2001-03

Analysis of the EPMIS data base

Short, William Baaclo; Bockenek, Jeffrey Mark.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/25671

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ANALYSIS OF THE EPMIS DATA BASE

by

William Baaclo Short

and

Jeffrey Mark Bockenek

September 1989

Thesis Advisor: Daniel R. Dolk
Thesis Co-advisor: Magdi Kamel

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b RESTRICTIVE MARKINGS
SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
DECLASSIFICATION/DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable): Code 54	7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000
NAME OF FUNDING/SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

TITLE (Include Security Classification): ANALYSIS OF THE EPMIS DATA BASE

PERSONAL AUTHOR(S): Short, William B. and Bockenek, Jeffrey M.

TYPE OF REPORT	TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ____ TO ____	1989, September	116

SUPPLEMENTARY NOTATION: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	EPMIS, INGRES

ABSTRACT (Continue on reverse if necessary and identify by block number)

The Emergency Preparedness Management Information System (EPMIS) has been developed as part of a research project with the Defense Communications Agency to build a decision support system for tracking national communications systems in times of emergency. The current EPMIS data base is implemented on the INGRES relational data base management system in a DEC MicroVax environment. The EPMIS program which interfaces with this data base operates at an extremely slow speed. In addition, documentation defining the structure and relationships within the data base is incomplete making it difficult to analyze and improve on its performance.

This thesis generates documentation for the EPMIS data base, including an entity-relationship diagram, in order to understand the logical structure of the data base. The EPMIS program is then analyzed to identify processing bottlenecks that degrade system performance. Modifications to the program are

DISTRIBUTION/AVAILABILITY OF ABSTRACT: [X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT [ ] DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION: Unclassified
NAME OF RESPONSIBLE INDIVIDUAL: Prof. Daniel R. Dolk	22b TELEPHONE (Include Area Code): (408) 646-2260; 22c OFFICE SYMBOL: Code 54Dk

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

U.S. Government Printing Office 1986—606-24

UNCLASSIFIED

#19 - ABSTRACT - (CONTINUED)

made to eliminate the bottlenecks and improve system performance.

Approved for public release; distribution is unlimited

Analysis of the EPMIS Data Base

by

William Baaclo Short
Lieutenant Commander, United States Navy
B.S., University of California at Los Angeles, 1977

and

Jeffrey Mark Bockenek
Lieutenant, United States Navy
B.S., Auburn University, 1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1989

ABSTRACT

The Emergency Preparedness Management Information System (EPMIS) has been developed as part of a research project with the Defense Communications Agency to build a decision support system for tracking national communications systems in times of emergency. The current EPMIS data base is implemented on the INGRES relational data base management system in a DEC MicroVax environment. The EPMIS program which interfaces with this data base operates at an extremely slow speed. In addition, documentation defining the structure and relationships within the data base is incomplete making it difficult to analyze and improve on its performance.

This thesis generates documentation for the EPMIS data base, including an entity-relationship diagram, in order to understand the logical structure of the data base. The EPMIS program is then analyzed to identify processing bottlenecks that degrade system performance. Modifications to the program are made to eliminate the bottlenecks and improve system performance.

TABLE OF CONTENTS

I. INTRODUCTION

The Emergency Preparedness Management Information System (EPMIS) is an automated tool which is being implemented to assist the Federal Government in the management of the Nation's Telecommunications Resources during times of national crisis or emergencies [Ref. 1:pp. ES 1-2]. EPMIS is being designed to provide timely, accurate, and relevant information concerning telecommunications capabilities. This information will support the roles of various managers tasked with maintaining the communication links across the country. INGRES is the relational data base management system used to implement the EPMIS data base.

A. BACKGROUND

The responsibility for preparing, coordinating, and maintaining a Federal Telecommunications Emergency Management Organization and plan falls on the Office of the Manager, National Communications System (OMNCS) [Ref. 2:p. 2-1]. In 1982, a national level exercise was conducted and the emergency telecommunications management and response procedures used by the NCS Emergency Preparedness organization were reviewed. It was at that time that the need for an automated decision support system to assist in the management and tracking of Federal telecommunications was identified. [Ref. 2:p. 2-2]

The EPMIS project was developed from the results of these studies and reviews, and is the information processing and decision support component of the National Telecommunications Management System (NTMS). The development was conducted in six phases beginning in March 1983 with the Functional System Description. A prototype for EPMIS was developed and was followed by the development of a portable system known as FAMIS. A portable unit, FAMIS was deployed to NCS regional managers. It was designed to perform similar functions as EPMIS but with reduced capabilities. The prototype EPMIS and FAMIS were tested during national level exercises in 1984. The exercises were used to represent the operational and feasibility test for the prototype system. [Ref. 2:pp. 2-2--2-3] The results of the exercise validated the concept of "automated telecommunications management support" [Ref. 2:p. 2-2]. The next phase was the development of a Full-Scale Integrated EPMIS/FAMIS which reached its Initial Operational Capability in 1986 [Ref. 2:p. 2-6]. The next development phase consisted of enhancements to the system design based on user interaction and input. An Integrated system with Full Operational Capability and system operation/maintenance are the final two phases in the development of the EPMIS project. The programming period for EPMIS began in FY87 and is due to be completed in FY91. The system is scheduled to attain full operational capability in 1990. [Ref. 2:pp. 2-4--2-6]

B. EPMIS SYSTEM DESIGN

The EPMIS system was designed to perform three different functions in support of national emergency and security issues. These include the following situations:

1. Localized regional emergencies such as floods and other natural disasters.

2. Emergencies affecting multiple regions of the nation that require national-level coordination, e.g., Three Mile Island incident.

3. Nationwide emergencies such as a potential nuclear attack. [Ref. 2:p. 4-2]

In order to support these functions, EPMIS was divided into three operational components:

1. National Level Component.

2. Regionally Deployed Component.

3. Regional Level Component.

The National Level component enables the monitoring, coordinating, and controlling of telecommunications during a national emergency. These duties would be performed under the auspices of the Manager, NCS. [Ref. 3:p. 2]

The Regionally Deployed component will be used for regional emergencies and/or multiple regions of the nation. It will be used for monitoring regional emergencies and for the coordination of actions which will affect multiple regions. [Ref. 3:p. 3]

The Regional Level component will provide the ability to manage information at a local level. This will enable regional/local managers to resolve local emergencies without

the direct involvement of the national level NCS. [Ref. 3:p. 3]

C. EPMIS DATA BASE

Two types of data bases are being maintained in support of EPMIS. The National Data Base will be used to collect and disperse data to the distributed data bases (discussed below) and can be used as a baseline to provide information on telecommunications resources available in the case of an emergency situation. Duplicate copies will be maintained at each region. These copies will be referred to as "shadow data bases." [Ref. 3:p. 3] A local data base will also be maintained at each region and will contain additional data which may be required by the regional managers to perform their mission at the local level. Data entered at the regional level can be used to update the National Data Base if appropriate. [Ref. 3:pp. 3-4] There are two sets of data which are available in the EPMIS Data Base. One set consists of government information and the other consists of industry information. Presently, only industry information on AT&T and MCI is available. [Ref. 3:p. 19] There are seven types of data contained in the EPMIS National Data Base consisting of the following:

1. Personnel.
2. Networks.
3. Nodes.
4. Links.

5. Operations Centers.

6. Assets.

7. Asset Centers. [Ref. 3:p. 19]

Data collection was undertaken in 1986 [Ref. 3:p. 19]. OMNCS has been trying to get the support necessary so that each member agency will be responsible for adding, updating, and maintaining their data in the data base. The NCS Data Base Administrator has the ultimate responsibility to ensure that the data are both accurate and complete. The Data Base Administrator also has the responsibility to provide the proper procedures for submitting and entering data to the appropriate federal and industry agencies. Specifications on data entry procedures and data submission procedures will also be provided by the Administrator. [Ref. 3:pp. 20-21]

D. SCOPE OF THE THESIS

Presently, the EPMIS program which interfaces with the data base operates at a very slow speed. It is often too slow to be used in a "real life" emergency and needs to be reworked, if possible, so that the program operates at a reasonable speed. The objective of this thesis will be to restructure the EPMIS data base design to improve the overall performance of the program.

The research will be limited to the analysis of the current EPMIS data structures and will include the construction of an Entity-Relationship model, the identification of processing bottlenecks within the EPMIS

program, and proposals for changes to the physical data base design that will reduce or eliminate the bottlenecks.

E. METHODOLOGY

The thesis will be performed in five steps which will be done in two separate sections. The first section will concentrate on the conceptual design of the data base and will consist of the following steps:

1. Generation of the Entity-Relationship models using the current data tables documented by the program developers.

2. Analysis of the logical structure of the EPMIS data base, which will include a description of the relationships between data groups.

The second section will deal with the physical design of the data base and will include:

3. Study of the physical structure of the EPMIS data base, including a description of INGRES data storage methods and how each data group is physically stored.

4. Analysis of the EPMIS program to identify possible bottlenecks within the system. This process will include identifying which modules access which tables.

5. Proposals for changes to the physical data base design that will reduce or eliminate the bottlenecks.

F. THESIS STRUCTURE

The remainder of the thesis will be structured as follows.

Chapter II presents a more comprehensive review of the EPMIS Data Base structure as well as a review of the EPMIS systems supported by the Data Base.

Chapter III analyzes the Logical Structure of EPMIS. Included is a review of the Entity-Relationship approach, Reverse Engineering, and the Entity-Relationship diagram generated.

Chapter IV reviews the Physical Structure of the Data Base and the various storage structures provided by the INGRES Data Base Management System.

Chapter V analyzes the EPMIS Program. The analysis covers the six modules within the EPMIS Program and concentrates on the response times for various queries.

Chapter VI presents specific recommendations to improve the efficiency of the EPMIS Program.

Chapter VII concludes the thesis by reviewing the intent of the thesis, summarizing the findings, and discussing the advantages of implementing the recommendations.

II. EPMIS DATA BASE FUNCTIONS

The information necessary to monitor and maintain the nation's telecommunications resources during times of national emergency or war is provided through the EPMIS Data Base. A key component in the EPMIS system, the Data Base not only provides information to each component level within EPMIS, but is also used to interact with many of the special access and other related programs which have a role in the overall management process. How this information is dispersed throughout the system and the support the Data Base provides to the various programs will be the focus of this chapter.

A. CENTRALIZED NATIONAL TELECOMMUNICATIONS DATA BASE (CNTDB)

The CNTDB, which is also known as the Master Data Base, is maintained by the National Level component of the EPMIS System. It is physically maintained at the NCS in Arlington, Virginia and provides the data for the National and Regionally Deployed EPMIS components. As the Master Data Base, it is considered to be the most current data base and is used to resolve any discrepancies in the updating of the information. In addition, it serves as the collection and dissemination point for data that resides on the other data bases deployed throughout the system. [Ref. 1:p 4-4]

As part of the EPMIS System, Shadow Data Bases are maintained at each of the regionally deployed component level sites and are designated as alternate sites for the Master Data Base in times of emergency. These data bases are exact duplicates of the Master and are used for ensuring EPMIS survivability, and to provide for independent regional operations in solving telecommunication problems at the lowest level possible. In addition, Regionally Deployed Data Bases are also maintained at the regional component level. These data bases not only provide the Regional Managers with national level data found on the Master Data Base, but also provide data which are specific to each region and that are not maintained at the other component levels. Examples of the type of data specific to a region include: Local commercial carrier Point of Contact information and Local NCS Member Agency Point of Contact information. [Ref. 2:pp. 4-5--4-7]

Acting as the baseline for telecommunication resources, the CNTDB is considered the National Data Base and provides the NCS with information on the resources which would be available during a national emergency. However, if an emergency occurs where the communication lines are broken, an alternate site (one of the Shadow Data Bases) may be assigned until such time that the communication lines are restored. It is for this reason that the information

maintained in the Shadow Data Bases must be consistent with the master. [Ref. 2:p. 4-4]

The updating of the information to the Master Data Base is accomplished in one of three ways: on-line individual transactions, batch processing, or batch updates received from the deployed data bases [Ref. 2:pp. 4-4--4-5]. Since the information in the data base is constantly being updated and revised, maintenance known as "copy cleanups" [Ref. 2:p. 4-5] is done on a periodic basis to the Master Data Base. This is done to ensure the efficiency of data access by removal of unusable space, and to assist in the elimination of data base corruption. [Ref. 2:p. 4-5]. After completion of each cleanup, both the Shadow Data Bases and the Regionally Deployed data bases will receive an entire copy of the Master Data Base. In addition, periodic updates are performed to ensure that each of the data bases deployed have current data. How often these periodic updates are accomplished may be determined by the "number of update transactions made to the CNTDB, a specific time period, or a combination of both." [Ref. 2:p. 4-5]

B. SPECIAL ACCESS AND/OR RELATED PROGRAMS

The various programs that interact with the EPMIS program will be reviewed in this section.

1. National Telecommunications Management Structure (NTMS)

This structure is being developed to "support the national security emergency preparedness (NSEP) telecommunication requirements" [Ref. 1:p. 5-1] which will be present during a national emergency or war. Within the structure there is a National Coordinating Center (NCC) as well as Regional Coordinating Centers located in each one of the seven telecommunication industry regions [Ref. 1:p. 5-1]. The NCC coordinates the telecommunications at the national level and the RCCs coordinate the NCC mission in their respective regions. Should the NCC be unable to provide the required NSEP support, one of the RCCs will assume the role of the national coordinator. The RCC designated will continue to perform its functions in the region assigned. Each of the RCCs has the capability to assume the role of the NCC. The basic concept of NTMS is to manage the resources below the national level (e.g., regional or local) whenever feasible. [Ref. 1:p. 5-1]

EPMIS is used as the Decision Support System for NTMS and "provides emergency telecommunications management information in support of NTMS operations." [Ref. 1:p. 5-2] EPMIS also provides information concerning those personnel vital to the operations of communication operations. This information is provided through several EPMIS functions and includes such items as emergency recall lists for NCS personnel, emergency points of contacts (phone and pager

numbers), and location information (home, work, and emergency) for designated personnel. Designed to support operations at the proper level, EPMIS eases the delegation of operations to the regional or local level when deemed appropriate. [Ref. 1:p. 5-2]

2. National Telecommunications Coordinating Network (NTCN)

A communication network which supports the NTMS, NTCN is "an integration of Public Switched Networks (PSNs) and Government communication networks, systems and facilities augmented pre-positioned dedicated equipment to provide a survivable and enduring network." [Ref. 1:p 5-2] The network is activated following the disruption of normal telecommunications connectivity. It would provide a "thin-thread communication link among the surviving NTMS locations requiring connectivity." [Ref 1:p. 5-2]

EPMIS is used in providing secure voice and data transfer between the NCC and RCCs via the NTCN. The NTCN facilitates communication between the EPMIS computers at the NCC and RCCs and should the NCC go down, also ensures connectivity between the RCC designated as the national level center and the remaining locations in the EPMIS system. [Ref. 1:p. 5-3]

3. Nationwide Emergency Telecommunications Service (NETS)

NETS "is based on the concept of increasing the connectivity between the surviving switches and transmission

facilities within the Public Switched Network (PSN) during and after national emergencies." [Ref. 1:p. 5-4] It provides the capability for low speed data communications by using those surviving switching and transmission facilities of commercial, private, and government networks [Ref. 1:p. 5-4]. In effect, NETS takes those routing options that are not normally used for PSN calls and pieces them together to form usable connections. This enables calls to be routed around those areas in the PSNs that have been damaged.

Since EPMIS does rely on the use of PSNs for some external data communications, it may be dependent on the use of NETS to provide that function. Information on which nodes in a particular network have call controllers, used by NETS to implement these new connections, can be maintained by EPMIS. EPMIS also has the ability to perform damage assessment on those nodes. [Ref. 1:pp. 5-4--5-5]

4. Expert Telecommunications Resource Allocation Module (XTRAM)

This module is an expert system enhancement to EPMIS which can be used in stand alone mode or interfaced with EPMIS. It provides the Resource Allocation Officer with "recommendations as to the desired allocation of the residual telecommunications resources in response to prioritized government requirements." [Ref. 1:p. 5-6] In addition, the system can provide a description of the decision process used in the generation of the recommendations.

EPMIS is used to provide the list of prioritized service requests and telecommunications resource information. However, problems do exist with this system. XTRAM does not provide the results of its analysis to the data base which results in the data base not being kept current. Also, "XTRAM was implemented on the DEC VAXstation and is unable to operate on the current Epmis hardware configuration" [Ref. 1:p. 5-7] as the shell software used by XTRAM cannot run on the DEC MicroVAX II minicomputer which supports EPMIS. Before XTRAM can be deployed, it must reflect the current hardware configuration used by EPMIS. Integrating XTRAM and EPMIS is the next step in XTRAM's development process and is currently underway. However, EPMIS uses INGRES as an information management tool and XTRAM does not. This will cause additional complications when XTRAM attempts to retrieve information from EPMIS. [Ref. 1:p 5-7]

5. <u>Telecommunications Service Priority System (TSP)</u>

TSP "is being developed to replace the current Restoration Priority (RP) system" and "will be used to process requests for assignment of provisioning and restoration priorities to new or existing NSEP telecommunications services." [Ref. 1:pp. 5-7--5-8] TSP MIS is used to provide automated support for the TSP system. It is a data base oriented application system and ensures

effective and timely priority assignment and monitoring of NSEP services by the TSP system. [Ref. 1:p. 5-8]

EPMIS must have access to the TSP system data base in order to properly prioritize the requests for the remaining telecommunication resources. EPMIS "can use the service priority assignments contained within TSP MIS to determine restoration priorities" and then uses this data "to determine which NSEP telecommunications service users should be given access to in the event of an emergency." [Ref. 1:p. 5-8]

6. Shared Resources (SHARES) High Frequency Radio Program

The SHARES network "will provide a backup capability to exchange critical information among Federal entities to support NSEP." [Ref. 1:p. 5-9] A communications infrastructure of federally controlled HF radio resources will be established and through this network, messages of critical importance will be passed among Federal agencies. SHARES can be used to pass information between EPMIS installations. Any classified information will require encryption prior to transmission via SHARES. [Ref. 1: p. 5-9]

EPMIS can be used to store information pertaining to SHARES stations. This would include primary and secondary transmitting frequencies, station locations, operating personnel, and assets. [Ref. 1:p 5-10]

C. EPMIS SOFTWARE PROGRAMS AND APPLICATIONS

The various software applications and programs used within EPMIS include both commercial application packages and those custom packages developed by the EPMIS designers.

1. INGRES

INGRES is the relational data base management system in which EPMIS is implemented. It includes the implementation of Query Language (QUEL) as well as application development tools. In EPMIS, INGRES will run on a Micro-VAX platform. [Ref. 3: p. 11]

INGRES/STAR is a distributed data base product which will enable different applications to efficiently access data across a variety of computer systems. "It provides universal access to information while at the same time allowing the local systems to maintain control of the security and integrity of local information." In EPMIS, INGRES/STAR will allow a PC at a regional site to access data which is located on a Micro-VAX and also will facilitate data transfer between Micro-VAXs located in different regions. [Ref. 3:p. 11]

INGRES NET coordinates the processing of an application program and the data base on two separate machines simultaneously. It is also used as a link between INGRES/STAR and data bases in remote locations. [Ref. 5:p. 10-11]

2. KERMIT

KERMIT is a protocol which is designed for the transfer of files over ordinary serial communication lines. This will allow EPMIS users to transfer files from a PC to a Micro-VAX and vice versa. It will also allow for storage of information on floppy disks and the transfer of data between locations. KERMIT will only be used for other than normal EPMIS communications. DECNET, INGRES/STAR, and/or INGRES NET will be used for normal communications. [Ref. 3:p. 12]

3. DECNET

Developed by Digital Equipment Corporation (DEC), "DECNET is a set of programs and protocols for use on DEC computer systems." [Ref. 3:p. 12] DECNET will be used to link the various MicroVAXs, which will be deployed throughout the country, to form a wide area network and will allow communications to occur between the regional Micro-VAXs as well as with the Micro-VAX located at NCS. It will also allow EPMIS users to use the VMS Mail and Phone utilities to communicate with other EPMIS users. [Ref. 3:p. 12]

4. PC Software Applications

The following applications will reside on the EPMIS PC and will provide the user with additional capabilities.

1. Desqview--a multiple tasking software program providing "windows" which allow the user to have more than one application operating at a time. [Ref. 3:p. 13]

2. Multimate--has been chosen as the word processing program on the EPMIS PC and Lotus 1-2-3 will be the spreadsheet application for EPMIS users. [Ref. 3:p. 13]

3. PC-DACS--a PC security system which will be required on the PCs. It will be used to ensure that only authorized users gain access to the system. This will be accomplished through the checking of usernames and passwords. It will also control data access by privileged users. [Ref. 3:p. 14]

III. ANALYZING THE LOGICAL STRUCTURE

A. BACKGROUND ON THE ENTITY-RELATIONSHIP APPROACH

Before starting an analysis of the logical structure of the EPMIS data base, it is useful to review the various elements that make up a data base and how a data base is designed. Much of the following descriptions are taken from Reference 4.

A data base is a collection of records which in turn is a collection of data items. For example, a record called STUDENT can contain data relevant to a particular student. A record is divided into several fields, or elements, which describe the data. NAME and GRADE are possible elements of the record STUDENT. As an example, "John Doe" may be the value of a data item described by the element NAME while "Sophomore" may be the value of a data item described by the element GRADE. These data items are also called attributes. A record is a collection of these attributes. A file is a collection of records of the same type. For instance, the file called STUDENT may be a collection of STUDENT records. A data base may contain many files and records. The process of organizing and storing these files and records is called data base design.

Data base design can be divided into two steps: logical design and physical design. "Logical data base design is

the process of designing the logical data structure for the data base." [Ref. 4:p. 4] Logical data base design involves analyzing the environment in which the data will be used, the data base system to be used, and the logical data structure types available in the data base system. "Physical data base design is the process of selecting a physical data structure for a given logical data structure." [Ref. 4:p. 4] Physical data base design involves the method used to store data onto physical data storage devices. Physical data structures are discussed in more detail in Chapter IV. In this chapter we concentrate on the logical structure.

"Currently there are few tools to aid the logical data base design process; the data base designer usually has to rely on intuition and experience. As a result, many data bases existing today are not properly designed." [Ref. 4:p. 4] Most data bases existing today were designed using the conventional data base design approach. This approach involves identifying the relevant data, then designing the logical structure into which the data will be organized and managed. There are two main problems with the conventional data base design approach. The first problem is that it is a complex task to accomplish. There are many things to consider, e.g., limitations imposed by the data base system on the way data can be structured, determination of access paths to each record, concern with the efficiency of updates and retrievals, etc. The second problem with the

conventional approach is that it is difficult for the designer to represent his logical description of the data base, called the schema, in a way that others can easily understand. This makes it hard for others to make changes or improvements on the data base. These two problems led to the development of the Entity-Relationship (E-R) approach by Professor Peter Chen in 1976.

As its title implies, the E-R approach involves "entities" and "relationships." "An entity is a 'thing' which can be distinctly identified." [Ref. 4:p. 17] There are many "things" in the real world. It is the responsibility of the data base designer to select the entities that are relevant to his data base (for the purposes of this discussion, entities can be thought of as files). A "relationship" is a connection or commonality between two or more entities. For example, MARRIAGE is a relationship between two person entities. The idea behind the E-R approach is to add an extra step between identifying the data and designing the logical structure. This middle step involves viewing the data from the "point of view of the whole enterprise" [Ref. 4:p. 9]. The description of this enterprise view is called the enterprise schema. "The enterprise schema should be a pure representation of the real world and should be independent of storage and efficiency considerations." [Ref.4:p. 9] This enterprise

schema is defined during the first phase of the E-R approach.

The E-R approach consists of two phases: constructing an E-R diagram to define the enterprise schema, and translating the enterprise schema into a logical structure. The E-R diagram, or E-R model, is the foundation of the E-R approach. The diagram consists of rectangles, which represent entities, connected by hexagons, which represent relationships. By taking the individual entities and determining the relationships between them, and then plotting the relationships onto an E-R diagram, the enterprise schema is developed. The enterprise schema shows the different groupings of related data items. This helps in getting a picture of the overall data base. The logical structure of the data base is derived from the enterprise schema. This is done by translating the E-R diagrams into data-structure diagrams. Developing data-structure diagrams involves determining whether relationships are one-to-one, one-to-many, or many-to-many, and analyzing the attributes of each entity to determine the key, which uniquely identifies each record in an entity, and the foreign keys which determine the relationship between entities. More than one data-structure diagram can be generated from the same E-R diagram. The data base system being used, along with the level of efficiency and interdependence desired, will determine the type of data-structure diagram. "The

logical data structure of data bases...can be expressed in terms of data-structure diagrams." [Ref. 4:p. 28] The E-R approach concludes after the development and implementation of this data-structure diagram.

The E-R approach has many advantages: (1) it simplifies and organizes the data base design process; (2) the enterprise schema is easier to design than the logical structure since it is not restricted by the capabilities of the data base system; (3) the enterprise schema is independent of the data base system, if it becomes necessary to change data base systems, the E-R diagram can serve as the basis for logical reconstruction of the data base; (4) it is a useful documentation tool since it is easy to get a grasp of the logical data base design by looking at the E-R diagram.

Because the E-R diagram makes it easier to understand the logical design of a data base, the analysis of the EPMIS data base will start with the construction of an E-R diagram. Since the logical data base design has already been developed, construction of the E-R diagram in this case will involve a reversal of the E-R approach. This process is referred to as Reverse Engineering.

B. REVERSE ENGINEERING: STEPS IN GENERATING THE E-R DIAGRAM

In order to perform an analysis of the EPMIS data base, an Entity-Relationship (E-R) model of the data base is

needed. This requires decomposing the data base to determine the entities, attributes, and relationships between the various entities. At the start of this thesis, only two documents were available that described the data base in any detail. The first document is the Data Dictionary generated by Booz-Allen Inc.--the prime contractor for the development of the EPMIS system. This dictionary defines all the tables in the data base and all the elements in each of the tables. The second document is a printout from the INGRES Data Base Management System (DBMS) which also describes all the tables in the data base. This second document provides information omitted from the Booz-Allen Data Dictionary. These two documents plus additional interviews with Mark Berman of Booz-Allen, provided the major sources for analysis of the EPMIS data base structure.

Using the two documents, construction of the E-R diagram can begin. The Reverse Engineering process involves four main steps: decomposing the data base, identifying the relationships, grouping the relationships, and generating the E-R diagram. After the E-R diagram is generated, an analysis of the logical data base structure can be done.

C. DECOMPOSING THE DATA BASE

An inspection of the data base shows that the data tables can be broken down into four categories: permanent, temporary, indexes, and views (although views are not

actually data tables, they are categorized as such for ease in breaking down the data base). The permanent tables are files that store data. The temporary tables are empty data tables which are filled only when information is to be printed out. These temporary tables are used to overcome the limitations of INGRES, which limits the number of line items that can be printed directly from the screen to ten. When the printing is done the information is erased from the temporary tables. Index tables effectively sort permanent data tables on different keys. The views are virtual tables not physically stored in the data base but derived from other data tables. Based on the original Booz-Allen Data Dictionary, there were 40 permanent tables, 25 temporary tables, 19 indexes, and six views. However, due to changes and updates, these numbers are no longer accurate. They are shown to give an idea of the size of the EPMIS data base. Exhibit 1 shows the grouping of the data tables.

Although the size of the data base indicates approximately 84 established tables, only the 36 permanent tables that are still active need to be analyzed. The temporary tables are only used for printing out information and therefore are not part of the entity-relationship structure of the data base. In addition, the indexes are auxiliary structures on permanent tables and therefore need not be treated as separate and distinct tables. Although views are logical representations of information from different

tables, EPMIS uses the views to merely make complex queries easier. As a result, the views do not represent real relationships in the data base and will not be considered. The process of determining the relationships within the EPMIS data base, and thus generating an E-R model of the data base, will be limited to the 36 permanent tables.

D. IDENTIFYING RELATIONSHIPS

The first step in determining what the entity-relationships are is to identify all the keys and foreign keys in each of the 36 permanent tables. Identifying these keys helps in making correlations between tables which, in turn, helps in determining relationships. Since the Booz-Allen Data Dictionary does not have any information concerning the keys, the INGRES DBMS report, along with information from Mark Berman, is the sole source for identifying the keys. Knowing the keys, it is a simple task to go back and identify all the foreign keys in each table.

The foreign keys tie together different tables that share common information. By identifying all those tables that are tied together, and by studying the description and purpose of each table, tentative relationships can be formulated. A simple example involves the two tables ASSET and ASSETCNTR. The ASSETCNTR table lists all the Asset Centers in the National Communications System along with information on the centers themselves. Its key is the element, asst_ctr_nam, which is also a foreign key in the

table ASSET. The ASSET table contains information on specific assets. It becomes apparent that specific assets listed in the ASSET table can be associated with the Asset Center it belongs to by taking the foreign key and crossing it into the ASSETCNTR table. Thus, by using the key attribute asst_ctr_nam to tie the two tables together, a relationship is established between ASSET and ASSETCNTR. Although establishing relationships between all 36 tables is much more complex, the basic process of using the foreign keys to tie tables together is the underlying method for establishing these relationships.

E. GROUPING THE RELATIONSHIPS

As relationships are established between all 36 permanent tables, four main groupings and two minor groupings emerge. Each group contains tables that are closely related. However, a grouping itself has basically no relationship with any of the other groupings. The four major groups are: Communication Networks, Damage Assessment, Service Requests, and Facility Requests. The two minor groups are: Emergency Activation Documents and Regional Situation. Exhibit 2 shows the breakdown by groups.

1. Communication Networks Group

The Network Asset Location grouping contains tables that describe each network in the National Communication Network including all resources such as assets, asset

centers, operation centers, personnel, links, and nodes that make up each of the various networks. In addition, the location of each of these resources is also described.

2. Damage Assessment Group

The Damage Assessment grouping contains tables that deal with determining the extent of damage to communication resources based on observed damage inputs. Certain tables in this grouping contain parameters that are used to determine the probabilities of destruction of certain communication resources and to predict the operational status of those resources.

3. Service Request Group

The Service Requests grouping contains tables that comprise all the information required to submit a service request to perform maintenance on communication resources. It also contains tables that store all the information on each service request, the status of each service request, and a journal or historical file of all previous service requests.

4. Facility Request Group

The Facility Requests grouping is similar in structure and function to the Service Request grouping except that its tables deal with requests to install new communication facilities.

As Exhibit 2 shows, four tables are shared between different groups: NETWORK, RESLOC, AGCYFUNCT and TSPRPMAP.

Although these tables are shared, no logical relationship exists between the different groups.

F. GENERATING THE E-R DIAGRAM

With the relationships between the 36 permanent tables determined, the final step is to plot the relationships onto an Entity-Relationship (E-R) Diagram. Reference 4 is the major source of guidance for generating the E-R Diagram. In addition, a software program entitled "ER-Designer" by Chen & Associates, Inc., was used to do the physical plotting of the model. Four separate E-R Diagrams correspond to the four major groupings. The diagrams show each table and the relationships between the tables (see Exhibits 3-6). Since the minor groups consist of only two tables each, an E-R Diagram will not be generated for them. Rectangles are used to represent the tables (entities) while hexagons are used to represent and describe the relationships. By viewing the E-R Diagrams it is much easier to understand the interactions and workings of the data base. Since the Booz-Allen Data Dictionary is the only document available for studying the data base, the E-R Diagrams will be extremely helpful in allowing closer analysis of the logical structure of the EPMIS data base. This closer analysis is required in order to discover inefficiencies in the performance of the INGRES Data base Management System.

G. ANALYZING THE LOGICAL STRUCTURE

By using the E-R diagrams, it is possible to perform an analysis of the logical structure of the data base. As the E-R Diagrams show, the data base is logically divided into four major groups and two smaller, independent groups.

1. Communication Networks

The first major grouping deals with EPMIS communication networks and the resources belonging to these networks. The relationships within this group revolve around two key tables, NETWORK and RESLOC.

The NETWORK table lists and describes all the various communication networks throughout the country. Each communication network contains the following resources: nodes (which are listed in the NODE table), links (LINK table), Asset Centers (ASSETCNTR table), Operation Centers (OPTRNCNTR table), and personnel who work on the network (PERSONNEL table). The tables that maintain these resources not only describe each resource, but also indicate to which specific communication network the resource belongs. These tables, therefore, have a direct relationship with the NETWORK table. The type of relationship they have is a many-to-one, i.e., a network can have many resources, but a specific resource can belong to only one communication network.

The other key table, RESLOC, also has a many-to-one relationship with all but one of the same tables as NETWORK.

In this case, each resource can have only one location, but one location can contain many resources. The one table that does not have a relationship with RESLOC is LINK. Links are defined by two different nodes situated at two different locations. As a result a link cannot be assigned a specific location and therefore has no relationship to RESLOC. The relationship between NETWORK and RESLOC is also many-to-one since a location may contain many networks while a network can only have one location. RESLOC has a relationship with one additional table, STATICLOC, which contains basically the same information as RESLOC. It is a static location table that is used mainly to help a user select a latitude/ longitude and horizontal/vertical combination based upon a user entered city and state. Its relationship with RESLOC is one-to-one.

The relationship between the ASSET table, which lists each unique asset in the system, and the ASSETCNTR table, is many-to-one. Each Asset Center can have more that one asset, but each asset is uniquely identified with one Asset Center. In the PERSONNEL table each person listed has a "personnel status" assigned indicating his availability. The allowable descriptions for this status element are listed in the PERSTATUS table. This means that when the status of a person is entered into the PERSONNEL table, the system verifies the entry with the PERSTATUS table to ensure the entry is a valid one. There is a one-to-many

relationship between the PERSONNEL table and the PERSTATUS table in that a specific "status" description can appear for more than one person, but an individual person can have only one "status" description. The NODE table and the LINK table are also related. Each communication link always consists of exactly two nodes. A node, on the other hand, can be a part of many links or, in some cases, not a part of any link. This type of relationship is many-to-many.

2. Observed Damage

The second major grouping deals with the observed damage to a communication facility. The key table in this grouping is the DMGOBSRVD table which contains information on the location and on the radii of destruction and impairment of the damage. Every table in this group has a relationship with the DMGOBSRVD table.

The tables LAYDOWN, RECTANGLE, and DMGEDRES are related to DMGOBSRVD in that each table provides additional information to the observed damage. LAYDOWN contains laydown information such as lat/lon of the observed damage, in addition to the coordinates, height of burst, and weapon yield. RECTANGLE provides information used in a damage assessment algorithm for determining the extent of rectangular damage. DMGEDRES contains the resources that have been predicted as being damaged. LAYDOWN and RECTANGLE have a one-to-one relationship with DMGOBSRVD while DMGEDRES has a many-to-one relationship.

Both STATEREG and DIRECTION have a one-to-many relationship with DMGOBSRVD. Both are static tables that not only contain the allowable values for certain elements in the DMGOBSRVD table, but also provide additional information on that element. STATEREG provides the state abbreviations for the state element along with the regions associated with that state and the latitudes and longitudes of an imaginary 'box' around each state. DIRECTION provides the compass heading for the direction element along with damage direction and angles used for calculating rectangular damage information. Values of STATEREG and DIRECTION can be used for more than one observed damage. However, an observed damage can use only one value of STATEREG and DIRECTION.

Since JDMGOBSRVD is an exact duplicate of DMGOBSRVD, there a one-to-one relationship between the two. JDMGOBSRVD serves as a historical record of all damage reports.

The DMGEDRES table has a many-to-one relationship with the RESLOC table. A location can contain many damaged resources, but a damaged resource can have only one location. Although this relationship provides a link with the first group of tables that deal with Communication Networks, there is no other logical relationship between the two groups.

3. Service Requests

The third grouping deals with the information required for an agency to request maintenance service on resources located at its facility. The key table in this group is the SERVREQUEST table.

The SERVREQUEST table contains all of the information related to a service request with the exception of additional comments. The table SERVCOMMENT contains the additional comments pertaining to that service request. The attribute ncc-number, which is a unique number by which NCC identifies service requests, is provided to the SERVCOMMENT table by the SERVREQUEST table. Their relationship is one-to-one.

The tables CARRIERS, AGCYFUNCT, TSPRPMAP, and REQSTATUS all have a one-to-many relationship with SERVREQUEST. Each table provides the allowable values for specific elements in SERVREQUEST. CARRIERS provides the abbreviated names of commercial carriers for the carrier_name element, AGCYFUNCT provides the list of agencies that may request service for the agency element, TSPRPMAP provides the Telecommunications Service Priority (or TSP) code for the tsp element, and REQSTATUS is a static table that provides the allowable status descriptions for the status element. A service request can have only one value from each of these tables, however, values from these

tables can appear on more than one service request, thus a one-to-many relationship exists.

The JOURNREQUEST table is just a historical record of all service requests. It contains the same elements and values as the SERVREQUEST table. Its relationship with SERVREQUEST, therefore, is one-to-one, and its relationship to the other tables is similar to SERVREQUEST. Similarly, JOURNCOMMENT is an exact duplicate of SERVCOMMENT and has a one-to-one relationship with both SERVCOMMENT and JOURNREQUEST.

The last table in this group is the FUNCTMAP table. It is related to the AGCYFUNCT table in that it contains the list of function codes and priorities that are assigned to an agency. These codes and priorities are used for prioritizing service requests. Since an agency can have only one function code/priority, while a function code/ priority can be assigned to more than one agency, there is a one-to-many relationship between AGCYFUNCT and FUNCTMAP.

4. <u>Facility Requests</u>

The fourth grouping of tables deals with an agency's request to establish a new facility. The structure of this group is similar to the Service Request Group. As a result, some of the tables used in the Service Request Group are also used in this group. The key table in this group is FACLTYREQ.

FACLTYREQ contains all the information involved in a facility request except for additional comments. FACLTYCOM contains the additional comments that apply to the facility request. FACLTYREQ and FACLTYCOM have a one-to-one relationship. There is also a one-to-one relationship between FACLTYREQ and CLAIMNO. CLAIMNO is a table that is used to generate a sequential number, called a claim number, that uniquely identifies a facility request. As a result, when a facility request is assigned a claim number this claim number can not be assigned to any other facility request. Thus, there is a one-to-one relationship.

FACSTAT, TSPRPMAP, AGCYFUNCT, and NETWORK all have a one-to-many relationship with FACLTYREQ. FACSTAT contains all the valid values for the status of a facility request. One of these values is assigned to the status element of FACLTYREQ. TSPRPMAP provides the TSP (or priority) code for the tsp element, and AGCYFUNCT provides the list of agencies that may make facility requests for the agency element. NETWORK provides the abbreviated name of the network that the new facility will be incorporated into. As with the other tables, the system checks to see that the abbreviation that is entered into the net_abbr_nam element of the FACLTYREQ table is valid by verifying it with the NETWORK table.

The JRNFACREQ table is a historical record of all facility requests and is an exact duplicate of the FACLTYREQ table. They have a one-to-one relationship. Likewise, JRNFACCOM is a duplicate of FACLTYCOM and has a one-to-one relationship with both FACLTYCOM and JRNFACREQ.

5. Emergency Activation Documents

The first of the two smaller groupings deals with emergency activation documents (ead's). This group consists of only two tables, EADLIST and EADS. EADLIST maintains ead's and their associated issue and rescind information. Each individual ead is identified by the ead_id element. EADS contains the text of each ead. The relationship between EADLIST and EADS is one-to-one.

6. Regional Situation

The final grouping deals with the functional status of each region in the nation. This group also consists of only two tables, STATNATN and SONSIT. STATNATN maintains the current situation for the nation and each region. The possible situations that can be used are obtained from the SONSIT table. Since a region can be in only one situation, and since a particular situation can apply to more than one region, the relationship between STATNATN and SONSIT is one-to-many.

With the analysis of the logical structure accomplished, the next step in analyzing the EPMIS data base is to examine the physical structure of the data base.

IV. PHYSICAL STRUCTURE OF THE DATA BASE

An analysis of the physical structure of the EPMIS Data Base consists of four parts: (1) reviewing the concept of physical structure and how it affects the speed of data retrieval and data storage capacity; (2) explaining how the INGRES Data Base Management System stores data; (3) describing the storage structures available in INGRES and the applicable situations in which they should be used; and (4) reviewing how the data in the EPMIS Data Base is actually stored. The last review will be done by looking at each table and identifying the storage structure currently used.

A. PHYSICAL STRUCTURE CONCEPTS

There are several areas of concern when determining which type structure to choose for storing data. Two of the major issues include access speed and disk space. Each structure offers certain advantages and disadvantages in these areas and it must be determined what type of data support is needed prior to the actual design of the data base. Some structures offer greater speed in accessing data, going directly to a record rather than scanning an entire table. Certain structures require more disk space than others because of the way the data is stored. It is important to fully understand these concepts and how they

will affect the performance of the data base once implemented. As the different storage structures available through INGRES are explored, these concepts will be further examined and the situations for which each structure is best-suited will be discussed.

B. INGRES METHODS FOR STORING DATA

In INGRES each table is stored as a file. Each file is then divided into pages based on the number of bytes of information. INGRES stores 2048 bytes to a page with 2008 bytes allowed for user data [Ref. 6:p. 3]. The remaining bytes are assigned as INGRES overhead. Each page is divided into records and the number of records per page is determined by the record width and storage structure. It should be noted that records cannot be split between pages. Pages become important because they become a factor in both access speed and disk storage requirements. For example, it takes as many disk I/Os (input/output transactions) to retrieve an entire table as there are pages in that table, and with a large table it is desirable to avoid scanning every page unless absolutely necessary. Scanning each page takes considerably more time than going directly to the particular page in which your data is stored. The ability to go directly to a particular page rather that scanning the entire table is a function of the storage structure and will be discussed in the following sections. [Ref. 6:p. 4]

C. INGRES STORAGE STRUCTURES

There are four storage structures available with INGRES. The characteristics of each storage structure will be reviewed along with the best situations in which to use that particular structure. A table is provided at the end of this section which ranks the best storage structure to use for different tasks.

1. Heap

The heap structure is the most basic of the four structure types. It is basically a default storage structure using sequential entry and access as its primary means for storing and retrieving data. The tables have no key columns so queries must scan every page in a table when retrieving data. Space is often wasted in this format because appends are placed at the end of the table, duplicate rows are not removed, and space from deleted records is not reused. This results in holes in the tables and wasted space. [Ref. 6:p. 9]

The heap size (number of records per page) is computed by dividing 2008 by the row width + 2 (tuple id) and the number of pages in the heap table is computed by dividing the number of records by the number per page [Ref. 6:p. 7]. These formulas calculate the amount of data that will be stored in a heap table.

A Heap table is best utilized in any of the following situations: (1) when loading a significant amount

of data for the first time as it is the fastest structure for appending data; (2) when the table has only a few pages; and (3) when queries are such that they always select the entire table (e.g., batch applications where every record must be processed). [Ref. 6:p. 10]

Situations in which a Heap table would not be very efficient include when: (1) fast access is required to one row or a subset of rows; (2) the tables are large; and (3) the need exists for unique keys (e.g., as would be needed in a Hash table). [Ref. 6:p. 10]

2. <u>Hash</u>

A Hash table is normally used for random accessing of records, but sequential access is also possible. Attempting to retrieve data without using the assigned key value or an exact key match will result in a sequential search of a Hash Table. However, the efficiency provided by storing records in a Hash table is lost if used for sequential access. Random or direct access is used to identify a specific record by its assigned key value alleviating the need to search the entire table. The key value (hash value) is determined by using a "hashing algorithm" which consists of nothing more than performing an arithmetic operation on a specified field within a record. The result of this operation is then used as the address for that record within the file. For example, in a personnel file, the employee number may be used to determine the

address of each record. Taking the last three or four digits of the employee number and then using that sequence of numbers as the record address is one addressing scheme [Ref. 7:p. 627]. This method is known as the "division/ remainder" [Ref. 7:p. 627] method and can use other number series to come up with the sequence (e.g., using the first, third, and fifth digit of the employee number). This scheme, however, may result in addresses that are not unique which will affect the performance of the hash table. Another hashing method known as "folding" splits the key into parts and then summing these parts, takes the total or a part thereof and uses that number as an address [Ref. 7:p. 627]. Folding can be combined with division/remainder to form an effective addressing scheme. [Ref. 7:p. 627]

The Hash table is structured with a number of main pages, each with a number of records assigned. The number of main pages is determined by the number of records in the table as well as the number of records which will fit on a page. The number of records on each page varies and is assigned to a main page based on the hashed value of its key. Should a main page become completely filled with records, an overflow page is used for any additional records assigned to that filled main page. In order to control the number of overflow pages needed, a fillfactor is established. The fillfactor determines how much space is left in each main page for additional records. The default factor

is 50% [Ref. 6:p. 14]. It is important to minimize the number of overflow pages because they result in duplicate keys and slow the processing time. It should be noted that the user can control the number of main pages and the fillfactor with the MODIFY command and that the number of main pages is fixed at the time of a modification. So, if a large amount of data is added which results in a number of overflow pages, a modification of the hash table will result in additional main pages and fewer overflow pages. [Ref. 6:p. 16]

As with the other structures, there are times when the Hash tables are the most efficient. When data are to be retrieved using exact key values, then the Hash table is the best storage structure to use. However, there are retrieval situations for which Hash tables are not effective: (1) when retrievals require range searches or pattern matching; (2) if retrievals use only a part of a multi-column key; (3) when you are required to scan an entire table; (4) when you join two tables without any restrictions; or (5) when you have many overflow pages after modifying a table but without many duplicate keys. [Ref. 6:p. 17]

3. Indexed Sequential Access Method (ISAM)

With the ISAM storage structure, either sequential or random processing can take place. When using sequential processing, the search can be started at the beginning of a file or at a particular record within a file. On the other

hand, random processing accesses a specific record which is located by the key value through the use of indexes. [Ref. 7:p. 620]

The ISAM table consists of "prime areas," "overflow areas," and "indexes" [Ref. 7:p. 620]. The prime area is used to store records/files and is the same as the main pages described in the previous section on Hash. The overflow area is used to place additional records in the table that are not able to fit in the prime area. This area equates to the overflow page in the Hash table. The third area called indexes is used to locate a specific record when using random processing [Ref. 7:p. 620]. ISAM searches a table using the indexes until it points to the desired location. It then uses a sequential search to find the particular record stored in that location. The number of queries needed for using this method is less than the number required for a search of the entire file. Using an ISAM table can be compared to using a dictionary to find the word "search." The process begins by using the index to access the "S" section of the dictionary and then doing a sequential search of the section until the word is found.

Unlike the Hash table, ISAM allows the use of range searches and pattern matching when accessing data. This method also has a fillfactor with a default value of 80% [Ref. 6:p. 19]. It should be noted that both the index and main pages are static after using the modify command and

therefore the table must be modified often if large amounts of data are to be inputted. [Ref. 6:p. 19]

Although similar in structure to the Hash table, ISAM uses a different access methodology and therefore is useful in somewhat different situations: (1) queries involving the use of range searches and pattern matching as described above; (2) a table that grows very slowly; (3) a large key; and (4) a table that is small enough so that frequent modification can still be performed efficiently. Times when ISAM would not be appropriate include: (1) when only exact matches are being done; (2) when a table is large and growing rapidly (use of binary tree would be more appropriate in this case); and (3) when the table cannot be modified on a regular basis. [Ref. 6:p. 20]

4. <u>Binary Tree</u>

The final structure that INGRES supports is the BINARY TREE (BTREE) table which has the same basic features as ISAM. The main difference is that with the BTREE the index is dynamic, meaning that as the amount of data increases so does the size of the index table [Ref. 6:p. 22]. The "BTREE table is a multilevel index that allows both sequential and direct processing of data records." [Ref. 7:p. 643] The table consists of two parts: (1) Sequence Set; and (2) Index Set. The Sequence Set allows for sequential access to data records by providing a list of pointers to each of the data records in the table. This is

in physical sequence and uses primary key values for accessing the records. On the other hand, the Index Set provides a rapid, direct access to records by providing an index which points to groups of entries in the sequence set. As the BTREE drops to a lower level, the seach area gets smaller. The structure of the BTREE is balanced, meaning that each data record is the same distance from the highest level of the index set. All data records that are in the table reside at the same level (lowest) of the BTREE. [Ref. 7:p. 643]

An example of how a record is accessed from a BTREE table is illustrated in Exhibit 7 [Ref. 6:p. 21]. In this example, the table consists of a set of records with numerical values ranging from 20 to 64. The records are indexed in sequential order based on their numerical value. A search for a specific record begins by assigning values to the highest level index. For this example, the record with numerical value 42 will be accessed. The highest level index consists of two pointers (links) assigned the values less than or equal to 35 and greater than or equal to 36 which splits the table in half. Note that a less than or equal to relationship points to the largest value on the page(s) and a greater than or equal to relationship points to the lowest value on the page(s) [Ref. 6:p. 21]. Pointers then lead to the next level where the assigned values cut the search area in half once again. This pattern continues

until a pointer is pointing to the record being accessed. In this case, the search for record 42 would continue another two levels before being accessed. A total of five accesses would be needed to retrieve record 42 by the the indexes and 19 accesses would be needed to retrieve record 42 by sequential search.

Binary Tree requires more overhead than does ISAM. This is because the index is more complex and it grows as the amount of data grows. Also the index does not decrease as data is deleted. The modify command must be used in order to shrink the index and to delete empty spaces within the pages which are caused by deletion of data. [Ref. 6:p. 22]

The Binary Tree table is most useful under the following circumstances: (1) the table is growing very quickly and has become too big to modify; (2) you require access through pattern matching and range searches but cannot afford to modify the table to ISAM; or (3) you will be joining entire tables to each other. On the other hand, you would not use Binary Tree when: (1) the table is static or growing slowly; (2) the key is large (it will have to be stored twice); or (3) when there are many users appending to the end of the table in a concurrent environment. This may cause a deadlock in the index as it grows larger. [Ref. 6:p. 23]

Table 1 [Ref. 6:p. 24] summarizes the above information and provides a ranking system to the storage structures available based on a variety of tasks. The rankings are as follows with 1 being the most desirable structure to use:

TABLE 1

RECOMMENDED STORAGE STRUCTURES

Task	Structure(s)
Bulk Loading Table w/ data	Heap - 1, BTree - 2
Removing duplicate rows	Hash , ISAM, or BTree - 1
Exact Match	Hash - 1, ISAM or BTree - 2
Range/Pattern Matching	ISAM or BTree - 1
Sequential Searches	Heap - 1, ISAM or BTree - 2, Hash - 3
Partial Key	ISAM or BTree - 1
Access to Sorted Data	BTree - 1
Joins on Large Tables	ISAM or BTree - 1
Index grows as Table grows	BTree - 1
Very Small Table	Heap - 1
Very Large Table (> 1 mil)	BTree - 1

D. EPMIS DATA BASE STORAGE

Each of the permanent tables, temporary tables, and indexes in the EPMIS Data Base is stored using one of the storage structures reviewed above. The initial question that must be answered concerning the data is whether or not it is stored in the manner that is the most efficient for its defined tasks. In order to determine this, the tables must be analyzed as to what their function is, what type of data is stored in the tables, how the data is stored in the tables, and what type of queries and updates are performed

on these tables. Most of the analysis of the data will take place with the analysis of the EPMIS program and will be reported in the next chapter. However, Exhibit 8 has been developed to facilitate the initial review of how each of the tables is stored. The exhibit is broken down by storage structure with each of the tables and/or indexes listed under the appropriate structure. The tables listed are permanent tables unless otherwise designated.

V. THE EPMIS PROGRAM

Improvements in the performance of the EPMIS program are contingent upon an understanding of what the program does and how it is organized. EPMIS is a menu driven program that allows the user to perform a number of different operations on the data base. The first screen the user sees upon entering the EPMIS program is the main menu. From this main menu the user has six categories from which to choose: (1) Emergency Activation Procedures; (2) Emergency Points of Contact; (3) Resource Management; (4) Damage Assessment; (5) Service Requests; and (6) Communications (see Exhibit 9). This discussion will only deal with the major modules/ submodules in the program.

A. EMERGENCY ACTIVATION PROCEDURES

This module allows the user to retrieve and display Emergency Action Documents (EAD). These documents are used to assist NCS personnel in emergency situations. Examples of EADs are: (1) Telecommunications Orders (TELORDS); (2) Telecommunication Instructions (TELINSTR); and (3) Presidential Executive Action Documents (PEADS). In addition, this module contains information on the current state of emergency in each of the ten federal regions and in the nation as a whole. A recall hierarchy of NCS personnel is also maintained by this module.

Upon entering this module, the user encounters a submenu with three options: (1) Emergency Action Documents; (2) Emergency State of the Nation; and (3) NCS Emergency Recall List.

1. Emergency Action Documents (EAD)

This submodule produces a table of all the EADs currently stored in the data base. For each EAD listed, the user can issue or rescind that EAD by entering the date of the issue/rescind action in the appropriate column to the right and then saving the newly entered data. The user can also display all the information pertaining to a specific EAD. In addition, the user can produce a printout of any of this information if so desired.

2. Emergency State of the Nation

This submodule allows the user to view and/or modify the time line, the type, and the situation description of the state of the nation. The time line indicates the "state of the nation" that a region or nation is in, i.e., NORMAL, PLAN D, etc. The type is an additional description of the time line, i.e., PRE, TRANS, POST, etc. The situation description describes the actual situation, i.e., DAY-TO-DAY OPERATIONS, ATTACK, NUCLEAR DISASTER, etc. This submodule produces a table that lists the status of the ten regions and of the nation as a whole. The user can make changes to any of this information directly on the screen.

3. NCS Emergency Recall List

This submodule allows the user to review the NCS Emergency Recall List. When entering this submodule, the initial display is the name of the first person on the NCS recall list and his phone number. The list of people that this first person is supposed to call can then be displayed along with the people they are supposed to call. In this way the user can cycle through the whole hierarchy of the Emergency Recall List. All the different phone numbers are listed for each person (autovon, commercial, pager, etc.) along with a description of his position.

B. EMERGENCY POINTS OF CONTACT

This module displays all the personnel who are designated as Emergency Points of Contact (EPOC). The personnel are presented one at a time in alphabetical order. In addition to the person's phone number, the address, region, building, and location (given in latitude and longitude degrees) are also shown.

C. RESOURCE MANAGEMENT

This module contains information on telecommunication resources. These resources are broken down into seven functional groups: (1) Networks; (2) Nodes; (3) Links; (4) Operation Centers; (5) Asset Centers; (6) Assets; and (7) Personnel. Resources within a group can be listed on the screen based on user specified parameters, i.e., all nodes

in the state of California, or all links that are currently down, etc. A specific resource can also be selected by the user and all the information on file for that resource can be retrieved for review or update. New resources can also be added to the data base via this module.

The first screen that the user sees upon entering this module is a menu with two options: (1) Enter Resources; or (2) Monitor Resources. Selecting either option will produce another menu which lists all seven functional groups. After a functional group is selected, a blank form with headings pertaining to that specific group is displayed. If the Enter Resources option is selected, the user can then enter the data directly onto the form. If the Monitor Resources option is selected, the user can either display all the resources or enter the parameters to display a specific list of resources. Each resource will be displayed one at a time and will include all the information pertaining to that resource.

D. DAMAGE ASSESSMENT

This module uses information inputted by the user concerning location and extent of damage from a nuclear attack or natural disaster, and predicts which telecommunications resources will be impaired or destroyed. The initial screen display is a menu with four options: (1)

Enter New Damage; (2) Execute Damage Information; (3) Monitor Damage Information; and (4) Review Journal Damage.

1. Enter New Damage

Selecting this submodule produces another menu. This menu lists the four types of damage information that can be entered: (1) Laydown Information; (2) Nuclear Damage; (3) Circular Damage; and (4) Rectangular Damage. Laydown information includes latitude/longitude coordinates, height of burst, and weapon yield. Nuclear Damage includes type of nuclear explosion, maximum range, etc. Both Circular and Rectangular Damage include city, state, direction, radius, height, width, and latitude and longitude in degrees, minutes, and seconds.

2. Execute Damage Information

This module gives the user the option of executing all damage information that has been entered to date, or executing just the newly entered damage information that has not been previously executed. Execution involves the processing of several mathematical algorithms in order to produce a list of resources that are predicted to be damaged or impaired.

3. Monitor Damage

This module produces a menu with three options: (1) Monitor Damage Observation; (2) Monitor Damage Resources; and (3) Damage Reports. Monitor Damage Observation will produce a table of damaged locations and

The table can cover either a specific region or the entire nation. Monitor Damage Resources will produce a list of resources that have been predicted damaged. The Damage Reports submodule allows the user to produce a printout of the Damage information.

4. Review Journal Damage

This module allows viewing of old damage reports that have been previously journaled. The user has the option of viewing all journaled damage reports or just viewing specific ones.

E. SERVICE REQUESTS

This module allows the user to record and manage claims for service and facility requests. Service requests are generated when an agency wishes service restored, or initiated in an emergency situation. Facility requests are generated when nodes and/or operating centers no longer provide vital communications. Each request is assigned a priority and is reviewed and managed by NCS. The priority is based on the function of the agency. As the national situation changes, the services are reprioritized. This module also enables journalizing of old service and facility requests.

Upon entering this module the user is presented with a submenu consisting of two options: (1) Manage Service Requests; and (2) Manage Facility Requests. Selecting the Manage Service Requests option produces another submenu with

three options: (1) Enter Service Request; (2) Copy An Existing Service Request; and (3) Review/Resolve Service Requests. Each of these options produces a form into which the user must enter the appropriate information. Selecting the Manage Facility Requests also produces a submenu with the following options: (1) Enter Facility Request; (2) Review/Resolve Facility Request; and (3) Review Journaled Facility Requests. Selecting any of these options will also produce a form into which the user must enter the appropriate information.

F. COMMUNICATIONS

This module provides communication between EPMIS users. There are three methods of communication available: (1) mail; (2) phone; and (3) messages. The mail method produces a preformatted message form that the user can fill in and have mailed. The phone method allows a modem hook-up with another user and permits interactive communication. The message method is similar to the mail method in that it provides non-interactive communication only. This module is only a communication service available to the EPMIS system user. It does not interact with the EPMIS data base at all. Consequently, this module will not be discussed in the analysis of the EPMIS data base.

Within the modules and submodules discussed above are numerous smaller submodules which contain code that manipulates the EPMIS data. During the processing of these

procedures and the manipulation of the data, bottlenecks are encountered which cause a significant delay in the processing of the program. The analysis in the next chapter attempts to pinpoint and resolve each bottleneck encountered in the EPMIS program.

VI. ANALYSIS AND MODIFICATIONS

The analysis of bottlenecks in the program will begin with time trials on each module and submodule of the program. The time trials consist of running each module/submodule of the EPMIS program three separate times and taking the average response time. The response time is the amount of time that elapses from the moment the "DO" or "ENTER" key is pressed until the information is presented on the screen. These time trials will be run using a DEC VAX minicomputer in a single user environment. Modules with response times of over five seconds are considered bottlenecks and will be analyzed for causes. However, there are many limitations to our ability to discover and correct bottlenecks, and as a result, alternatives to these methods will need to be discussed.

A. LIMITATIONS AND ALTERNATIVES

The EPMIS program and data base being used for analysis are also being used by Roland and Associates, a private business subcontracted to develop the Damage Assessment portion of the EPMIS system. Because of this, our access to many of the modules and operations in the program is restricted. Since many of the modules have restricted access, time trials cannot be performed on them. However, these modules may still have operations which can cause

bottlenecks. As a result, many of the bottlenecks in these restricted modules will have to be identified by analyzing their code. This will involve close to 200 procedures with an average of approximately 250 lines of code per procedure. One way of making this task easier is to first analyze those bottlenecks that can be immediately identified via time trials. By determining the causes for these bottlenecks and recognizing the situations that can lead to a bottleneck, reviewing the code can be accomplished by looking specifically for those situations with potential for creating a bottleneck. Using "modular testing," separate time trials on code determined to have potential bottlenecks can then be run, thereby allowing identification of actual bottlenecks without needing to perform time trials on those program modules that have restricted access.

Modular testing will not only be used for discovering bottlenecks in modules that have restricted access, but also for finding solutions to fix the bottlenecks. Changes to the EPMIS program or to the EPMIS data base cannot be done, because both the program and the data base are also being used by Roland and Associates. Therefore, to determine whether a proposed solution will work, modular testing will be required. To perform modular testing, it is necessary to isolate the particular process that is causing the bottleneck and determine which data tables are being utilized by the process. Then by creating a separate and

"local" data base with the desired data tables included, the process can be duplicated and run against the local data base. Changes can then be made to the code or to the data base structure without affecting the actual EPMIS system. The system is not affected since changes are now being done within the local data base and not within the EPMIS data base. This is referred to as modular testing since the testing is being performed on only a small part or "module" of the EPMIS program and is separated from the rest of the program. Through modular testing, bottlenecks in those modules that have restricted access can be identified, and solutions to fixing the bottlenecks can be tested. Even though it can only be shown that the solutions work on the modules, there is no reason to suspect that these same changes will not also work when made to the actual EPMIS system.

There are two types of bottlenecks that this thesis will not resolve: (1) those caused by the operating system; and (2) those caused by security access validation procedures. The print function in many modules involves retrieving information to be printed and then invoking an operating system procedure to do the actual printing. Although the process of retrieving the information can be analyzed for bottlenecks, the performance of the operating system in printing this information is beyond the scope of this thesis. Therefore, bottlenecks that are caused by operating

system processing will not be resolved. In addition, the EPMIS system has special procedures and data tables that are used strictly for checking and verifying user names, passwords, and authorized access to certain data tables. Not only will INGRES not allow us to view these data tables, it will also not allow us to access the code. Therefore, bottlenecks that are caused by procedures verifying access clearance also will not be resolved. Although these types of bottlenecks cannot be resolved here, solutions proposed in the resolution of other accessible bottlenecks may work here as well if implemented by someone with authorized access to the modules and data tables.

Identification of those modules and submodules with slow response times leads to an analysis to locate the cause of the bottlenecks. By identifying the causes, methods to eliminate them can be proposed that will significantly improve the performance of the EPMIS program. In order to analyze how the processing takes place within the program, however, it is necessary to understand some of the capabilities and limitations of the INGRES Data Base Management System.

B. OPTIMIZATION FEATURES WITHIN INGRES

As discussed in Chapter II, INGRES is a relational Data Base Management System (DBMS) that stores data as tables. Retrieval of data requires a search by INGRES for the applicable table. Included in the INGRES data retrieval

mechanism is a subcomponent called an "optimizer." The function of the optimizer is to "choose, for each query it processes, an optimal access strategy for implementing that query." [Ref. 8:p. 25] With INGRES, when the user requests data, he does not have to be concerned with where the data is or how the data is accessed. This is left to the INGRES optimizer. The optimizer will determine the quickest and most efficient access strategy for retrieving the data. The access strategy attempts to avoid sequential searches of large data files by using keys and indexes to rapidly locate data. The strategy that the optimizer uses is referred to as the access path. It is important to note that the optimizer can only utilize access paths (e.g., indexes) that are available or have been established by the user within the data base. If the user does not create efficient access paths, the optimizer will have no choice but to use whatever path is available. Some ways that the user can create efficient access paths are by changing the physical storage of the data, creating indexes, or even combining tables to eliminate expensive joins. However, any change to the data base to improve data retrieval efficiency may affect the processing efficiency of some other process such as data input. It is up to the user to determine what the tradeoffs will be.

INGRES has another optimization feature that doesn't require making tradeoffs. It is the OPTIMIZEDB command.

When the optimizer determines the optimal access path to take, it refers to the INGRES Data Dictionary to determine what access paths are available. The INGRES Data Dictionary is a "repository for information...concerning various objects that are of interest to the system itself." [Ref. 8:p. 103] Information on such objects as tables, views, indexes, etc., are maintained in the Dictionary. Some of the information is always kept up to date; others are "updated only on request, because the overhead of maintaining them continuously would be too great." [Ref. 8:p. 235] The OPTIMIZEDB command is the method for requesting that all the information in the Dictionary be updated. This command should be utilized whenever "a significant amount of update activity occurs on a given data base." [Ref. 8:p. 235] Since there are no additions, deletions, or changes made to the data base, OPTIMIZEDB is one method for improving the retrieval efficiency of a program without degrading the efficiency of other processes within the same program.

Another method for improving retrieval efficiency is to determine whether the storage structure used for each table is the most efficient. Chapter IV discussed the various storage structures available with INGRES and under what conditions each storage structure is most efficient. By studying the characteristics and uses of each data table, the most efficient storage structure can be determined. If

the actual storage structure is different, the performance of the program can be improved by changing it to the more efficient structure.

Other methods available within INGRES for improving program performance are: (1) compressing data which saves disk space and can improve performance; (2) specifying the fill factor for data pages, a hash storage scheme does not work well if the pages are filled close to 100% capacity, a 50% fill factor is normally best; and (3) utilizing EQUEL, a high-level language consisting of QUEL statements embedded into a programming language (such as C, FORTRAN, Pascal, etc.). EQUEL has capabilities not available with QUEL and can be used in conjunction with QUEL statements. Aside from the use of EQUEL, these methods will not normally cause a significant improvement in efficiency. They should normally be used to "fine tune" the performance of a program that is already fairly efficient.

C. THE ANALYSIS PLAN

With this knowledge of some of the capabilities and limitations of INGRES, a plan on how to look for processing bottlenecks and how to fix them can be developed. Three major areas will be looked at: (1) the logical structure of the data base; (2) the physical structure of the data base; and (3) the program code. When looking at the logical structure, the main emphasis will be on finding "join" operations. "Join" operations involve the joining of two or

more different data tables. They usually occur when processing a retrieval command which has a search clause involving more than one table. These joins are generally expensive to process. By changing the logical structure of the data base (i.e., combining tables, adding/deleting elements) joins can be minimized. When looking at the physical structure we will analyze the retrievals that are done on those data tables that have been identified as being involved in a bottleneck. The analysis will include determining what access path the optimizer is using to make the retrieval and to see if a more efficient access path can be created. Analyzing the access path will involve studying the storage structure used by those data tables that are involved with bottlenecks and determining if the structure is the most efficient. The physical structure of the data base will be the major area of concentration in resolving bottlenecks. When looking at the program code, a review of the application program will be done to determine if more efficient programming can be used to take advantage of the optimizer and the storage methods used. By using these three major areas as a guide, analysis of the bottlenecks can start. The procedure names of executable files which contain the bottleneck will be used to identify the location of the bottleneck within the EPMIS program.

D. PROGRAM ANALYSIS

1. Emergency Activation Procedures Module

There are two major bottlenecks in this module, one which occurs in the three submodules, and one which occurs in the NCS Emergency Recall List submodule. The bottleneck that occurs in the three submodules involves the print function in which temporary tables are used. When the print operation is executed the first process that occurs is to load the temporary table with the data to be printed out. This is done by using the "unloadtable" command to extract the data from the table field (the information that is on the screen and which is to be printed out), and then using the "append" command to load the extracted data into the temporary table. When this is done a "call system" command is made to the operating system to print out the information that is in the temporary table.

The average time for the print function to process before printing starts is 36.82 seconds. An analysis of the "unloadtable" and "append" commands shows that both transactions together take less than three seconds. The delay apparently occurs when the "call system" command is made. Therefore, the bottleneck is caused by the operating system and not the data base management system. As previously discussed, this type of bottleneck will not be resolved by this thesis effort. It is presented here to show how the conclusion is reached that the operating system

is the cause of the bottleneck in the print operations. This same type of delay is seen in all the print functions throughout the EPMIS program. Analysis shows that the cause is the same in all of them. Therefore, the print function bottleneck will not be discussed further.

The second bottleneck in this module involves the submodule NCS Emergency Recall Lists which makes numerous accesses to the PERSONNEL table. Each access involves retrieving data from the table based on selection criteria. For example, in the procedures 'erecall1' and 'egetrcrp1,' there is a retrieval command for all personnel whose POSITION attribute has the value "NCS/DCAOC":

```
retrieve (personnel.position,
          personnel.pers_name,
          personnel.ddd_phone,
          personnel.fts_phone,
          personnel.von_phone,
          personnel.pager)
 where personnel.position = "NCS/DCAOC" .
```

There is also a separate retrieval command for all personnel whose POC_RECLDBY attribute also has the value "NCS/DCAOC":

```
retrieve (pos = personnel.position,
          personnel.pers_name,
          personnel.ddd_phone,
          personnel.fts_phone,
          personnel.von_phone,
          personnel.pager)
 where personnel.poc_recldby = "NCS/DCAOC" .
```

The average processing time for each of these submodules, with both retrievals in it, is 19.15 seconds. Separate time trials on each retrieval command shows that

the retrieval using the POSITION attribute as the selection criterion takes less than two seconds while the retrieval using the POC_RECLDBY attribute as the selection criteria takes an average of 17.46 seconds. The bottleneck is obviously with the POC_RECLDBY retrieval. To find out why this bottleneck exists, a determination of the access path that the INGRES optimizer is taking to access the data is required. This involves studying the storage structure and the indexes of the PERSONNEL table.

The PERSONNEL table uses a binary tree storage method with the elements (attributes) PERS_NAME and STATUS as its keys. There are also four indexes on the PERSONNEL table: (1) X1PERSONNEL, which uses a binary tree structure with POSITION as the index key; (2) PERSNET, binary tree with index keys NET_ABBR_NAM and PERSONL_INFO; (3) PERSLATLON, binary tree with index keys LATITUDE and LONGITUDE; and (4) PERSDMG, isam with index key DMGCNT. Since the index X1PERSONNEL has POSITION as its index key, the INGRES optimizer uses this index when performing a retrieval operation on the PERSONNEL table with POSITION as the selection criteria. As a result, the retrieval is performed rapidly. Since there are no indexes with POC_RECLDBY as the index key, the optimizer must do a sequential search of the entire PERSONNEL table when performing a retrieval operation with POC_RECLDBY as the selection criteria. Since the PERSONNEL table currently

holds 904 rows of information this retrieval takes a much longer time, resulting in a processing bottleneck. The solution to this bottleneck is to create a new index on the PERSONNEL table with POC_RECLDBY as the index key. To test this solution, the index needs to be established and new time trials taken. As stated above, a retrieval against the PERSONNEL table with POC_RECLDBY as the selection criteria takes an average of 17.46 seconds without the index. After the index is created the average time for a retrieval is 1.56 seconds. This is a 91% improvement in processing efficiency. By creating an index on the PERSONNEL table with POC_RECLDBY as the index key, the processing time for the Emergency Recall List submodule can be reduced from an average of 19.15 seconds to an average of 3.25 seconds.

2. <u>Emergency Points of Contact (EPOC)</u>

This module consists of only one procedure, 'pemgypoc1,' and has an average processing time of 29.29 seconds. As with the NCS Emergency Recall List submodule, this module also accesses only the PERSONNEL table, and produces a bottleneck when retrievals are made against the table. However, when the retrieve command is issued in this case, the search criterion is any person whose POSITION attribute has the letters "EPOC" appearing anywhere in the attribute:

```
retrieve (personnel.position,
          personnel.pers_name,
          personnel.ddd_phone,
          personnel.fts_phone,
          personnel.von_phone,
          personnel.pager)
 where personnel.position = "*-EPOC*"
 sort by pers_name
```

Since there already is an index with POSITION as the index key (X1PERSONNEL), it seems that the optimizer would use this index for retrieving the data, and that the retrieval process should be much faster. However, since the search criterion is for a series of letters that could appear anywhere in the title (vice only in the beginning), the INGRES optimizer realizes that the index will not help in this case and therefore performs a sequential search on the base table. To improve the performance, therefore, the logical structure of the table will have to be adjusted. By adding a new element called EPOC to the base table, this element can be used to designate those personnel who are EPOCs, i.e., if the person is an EPOC the EPOC attribute will have a value of "y," if not the attribute will be blank. By then creating an index on the PERSONNEL table with the new element EPOC as the index key, a retrieval for all personnel who are EPOCs will process in less than three seconds eliminating a major portion of the bottleneck. Although this is a big cause of the bottleneck, it is not the only contributor.

By running separate time trials for each command issued against the PERSONNEL table, we discovered that

another major contributor to the bottleneck is the "sort" command which takes an average of 10.8 seconds to process. The "sort" command is used after the retrieve command to put the EPOC names in alphabetical order. However, by studying the physical structure of the PERSONNEL table, we discovered that this "sort" command is unnecessary. Since the PERSONNEL table uses a binary tree storage structure with PERS_NAME as the key, a sequential retrieval of the data from the table would be in PERS_NAME (alphabetical) order. As previously mentioned, when the retrieve statement is executed, the optimizer performs a sequential retrieval against the base table. The result, therefore, is already in alphabetical order eliminating the need for the "sort" command. Taking out the "sort" command reduces the processing time for the module from an average of 29.29 seconds to 18.49 seconds. A comparison between the output of a retrieval with the "sort" command and the output of a retrieval without the "sort" command shows that the outputs are exactly the same.

As discussed above, this module has two bottleneck solutions: (1) adding the EPOC element to PERSONNEL and creating an index with EPOC as the key; and (2) eliminating the "sort" command. These two solutions, however, cannot be implemented together. The problem is that if the EPOC index is created and used by the optimizer to retrieve the data, the output will no longer be in alphabetical order. Thus,

the "sort" command will still be required and an 18 second bottleneck will still exist. The way to get around this problem is to add the PERS_NAME element to the EPOC index, i.e., create an index with EPOC as the primary sort key and PERS_NAME as the secondary sort key, e.g.,

index of personnel is xperson (epoc, pers_name).

This will put the EPOC personnel in PERS_NAME order so that when the retrieval of EPOC personnel is made the output will be in alphabetical order. Consequently, the "sort" command can now be eliminated. By creating this index with EPOC and PERS_NAME as the keys and eliminating the "sort" command, the processing time for this module can be reduced from an average of 29.29 seconds to under four seconds, an 86% improvement in processing efficiency.

3. Resource Management

This module has a number of bottlenecks, many of which are caused by the same transaction. For example, there are 11 procedures which use the exact same transaction: (1) uvalidloc; (2) mupdpers1; (3) mupdoc1; (4) mupdnode1; (5) mupdnet1; (6) mupdac1; (7) maddpers1; (8) maddoc1; (9) maddnode1; (10) maddac1; and (11) maddnet1. The transaction is a retrieval of data from the RESLOC table, which contains information on the location of various telecommunication resources. The retrieval is requested

using six selection criteria: (1) LAT_DEGREES; (2) LAT_MINUTES; (3) LAT-SECONDS; (4) LON_DEGREES; (5) LON_MINUTES; and (6) LON_SECONDS:

```
retrieve (latitude = resloc.#latitude,
          longitude = resloc.#longitude)
 where resloc.#lat_degrees = lat_deg
    and resloc.#lat_minutes = lat_min
    and resloc.#lat_seconds = lat_sec
    and resloc.#lon_degrees = lon_deg
    and resloc.#lon_min = lon_min
    and resloc.#lon_seconds = lon_sec .
```

A review of the resloc table shows that it uses a hash structure with LATITUDE and LONGITUDE as the keys. There are no secondary indexes. Since there are no indexes that can used to access the data, the optimizer must perform a sequential search of the data table. Since this table contains 784 rows the transaction can take some time. However, since the number of pages is small (113 as compared to 346 for the PERSONNEL table), the transaction by itself only takes 4.47 seconds. Unfortunately, not only does this transaction appear in 11 procedures, it also appears twice in all but 'uvalidloc.'

The second time this type of transaction appears within a procedure, it is in the form of a "delete" command. The transaction involves deletion of data from the RESLOC table that matches the selection criteria:

```
delete resloc
 where resloc.lat_degrees = lat_deg
    and resloc.lat_minutes = lat_min
    and resloc.lat_seconds = lat_sec
    and resloc.lon_degrees = lon_deg
    and resloc.lon_minutes = lon_min
    and resloc.lon_seconds = lon_sec
```

Although the retrieve and the delete are two different commands, the optimizer performs the same operation in both. This means that the optimizer must perform another sequential search of the table to the find the data to delete. As a result, to process entirely through one procedure will take at least 8.94 seconds. Since more than one of these procedures may have to be called in order to process one module/submodule, the total transaction time can become very long. As before, the solution to the bottleneck is to create an index on the RESLOC table with LAT_DEGREES, LAT_MINUTES, LAT_SECONDS, LON_DEGREES, LON_MINUTES, and LON_SECONDS as the keys. With this index, processing time is reduced to 2.16 seconds. However, since this transaction is repeated a number of times, a faster transaction time is desired. When an index is created, INGRES automatically uses an isam structure for the index. In this case, though, since the selection criteria is an exact match of the index keys, use of a hash structure will result in a faster retrieval. When the index is modified from isam to hash the transaction time is reduced to 1.73 seconds. The decrease from 4.47 seconds to 1.73 seconds is a 61% improvement in processing efficiency and becomes significant when this retrieval is performed frequently.

There are two additional procedures that also access the RESLOC table in a similar fashion, and may cause

bottlenecks when combined with other procedures. Both 'rcombine' and 'mcombine' process an append command that takes data from the RESLOC table and appends it to a temporary table. This append command appears twice in each procedure. The append is issued using two selection criteria:

```
range of l is resloc
append to combloc (lat_degrees = l.lat_degrees,
                   lat_minutes = l.lat_minutes,
                   ... etc. )
   where l.#city = city and l.#state = state .
```

As with the retrieve and delete commands, the optimizer must perform a sequential search of the RESLOC table to find the data since there are no indexes that can be used. A transaction time of 4.49 seconds is reduced to 1.58 seconds with the creation of a new index using CITY and STATE as the keys, a 65% improvement in processing efficiency. Since the append transaction appears twice in each procedure, this bottleneck is reduced from 8.98 seconds to 3.16 seconds.

The major source of bottlenecks in the Resource Management module is the result of "join" transactions. A join occurs when data is retrieved using selection criteria that involves more than one table. For example, retrieval of information on a person whose location appears in both the PERSONNEL table and the RESLOC table will require a join of both tables. A join can be viewed as taking the cartesian product of both tables and then deleting those rows that do not meet the selection criteria. This means

that if one table has three rows and the second table has four rows, the cartesian product will produce a virtual table with 12 rows from which those rows that do not meet the selection criteria will be eliminated. It is easy to see how joins can become very time consuming if just one of the tables is large. The cartesian product of the PERSONNEL table (904 rows) and the RESLOC table (784 rows) will result in a virtual table of 708,736 rows. This is compounded if additional tables are included in the selection criteria requiring additional cartesian products. Although the INGRES optimizer will not necessarily perform a cartesian product to process every join, the processing time is still significantly affected by the size of the tables being joined. When performing joins of many tables, the optimizer attempts to find a sequence that will produce the smallest number of searches. It will attempt to process the most restrictive selection criterion first, thereby minimizing the number of rows to join. Although there are joins appearing in a number of procedures, the ones that cause a bottleneck in this module involve the use of views.

There are six procedures that utilize a view: (1) mlstpers1 (uses view LISTPERS); (2) mlstasst1 (uses view LISTASSETS); (3) mlstnet1 (uses view LISTNET); (4) mlstnode1 (uses view LISTNODE); (5) mlstoc1 (uses view LISTOC); and (6) mlstac1 (uses view LISTAC). The reason why joins in these procedures produce bottlenecks is because of the

number of joins involved in creating the view. For example, the code that defines the view LISTNODES is as follows:

```
range of n is node
range of l is resloc
range of rv2 is network
range of s is statereg
define view listnodes (
     node_name = n.node_name,
     net_abbr_nam = n.net_abbr_nam,
     ccf_indctr = n.ccf_indctr,
     ccm_indctr = n.ccm_indctr,
     lat_degrees = l.lat_degrees,
     lat_minutes = l.lat_minutes,
     lat_seconds = l.lat_seconds,
     lat_direct = l.lat_direct,
     lon_degrees = l.lon_degrees,
     lon_minutes = l.lon_degrees,
     lon_seconds = l.lon_seconds,
     lon_direct = l.lon_direct,
     latitude = l.latitude,
     longitude = l.longitude,
     state = l.state,
     region = s.region,
     a_status = n.act_status,
     p_status = n.pred_status,
     agency = rv2.agency)
   where (n.longitude = l.longitude)
     and (n.latitude = l.latitude)
     and (n.net_abbr_nam = rv2.net_abbr_nam)
     and (s.st_abbr_name = l.state)
```

As can be seen, the creation of LISTNODE involves the joining of four tables. Although to the user, a view can be treated as an actual data table, in reality it is merely a collection of data from different tables. However, the actual creation of the view is not made until the view is named in a retrieval operation, like in the procedure 'mlstnode1':

```
nodelist = retrieve
        (listnodes.node_name,
         listnodes.net_abbr_nam,
         listnodes.ccf_indctr,
         listnodes.ccm_indctr,
```

```
          listnodes.lat_degrees,
          listnodes.lat_minutes,
          listnodes.lat_seconds,
          listnodes.lat_direct,
          listnodes.lon_degrees,
          listnodes.lon_minutes,
          listnodes.lon_seconds,
          listnodes.lon_direct,
          listnodes.pred_status)
        where listnodes.node_name = sel_node_name
          and listnodes.net_abbr_nam = sel_net_name
          and listnodes.a_status = sel_status
          and listnodes.agency = sel_agency
          and (listnodes.state = sel_scope or
               listnodes.region = sel_region)
```

When processing this retrieval, the optimizer combines the selection criteria for creating the view LISTNODE with the selection criteria for this retrieval and processes them as one transaction. As a result, the optimizer actually processes the following retrieval:

```
     range of n is node
     range of l is resloc
     range of rv2 is network
     range of s is statereg
     retrieve (
          n.node_name, n.net_abbr_nam,
          n.ccf_indctr, n.ccm_indctr,
          l.lat_degrees, l.lat_minutes,
          l.lat_seconds, l.lat_direct,
          l.lon_degrees, l.lon_minutes,
          l.lon_seconds, l.lon_direct,
          l.pred_status)
        where n.longitude = l.longitude
          and n.latitude = l.latitude
          and n.net_abbr_nam = rv2.net_abbr_nam
          and s.st_abbr_name = l.state
          and n.node_name = sel_node_nam
          and n.net_abbr_nam = sel_net_nam
          and n.act_status = sel_status
          and rv2.agency = sel_agency
          and (l.state = sel_scope or
               s.region = sel_region)
```

The values of sel_node_nam, sel_net_nam, sel_status, sel_agency, sel_scope, and sel_region are determined by the

user. If the user wants only those nodes that are in region 1 he inputs a "1" for sel_region. If the user wants the above information retrieved on all nodes, he inputs "*" for every selection. As mentioned above, the optimizer attempts to process the most restrictive selection criterion first. Therefore, if the user wanted only the information retrieved for a specific node (e.g., ISIC), the optimizer would first process the criterion "n.node_name = sel_node_nam" (where sel_node_nam equals "ISIC"). This would produce just one data item, greatly simplifying the remaining joins, and resulting in a fast retrieval. If the user wanted information on all nodes in region 1, and there were only a few nodes in that region, the optimizer would process the criterion "s.region = sel_region" first (where sel_region = 1), again producing a fairly rapid retrieval time. If, however, the user wanted all the information on all the nodes, there would be no selection criterion that would be very restrictive. As a result, the joins would become very complex because of all the data manipulation, resulting in a long retrieval time.

The processing time for each procedure that uses a view is directly related to the size of the data tables that the view accesses. Views that use large tables, such as LISTNODES, produce transaction times of up to 56.2 seconds while views that use smaller tables, such as LISTNET, produce shorter transaction times down to 6.31 seconds (this

is assuming the user wants a retrieval of all data in the view). To resolve this bottleneck, we need to break the retrieval into two types: (1) retrievals of all data, i.e., all "sel_" values equal "*"; and (2) retrievals of selected data, i.e., sel_region = 1.

If selected data is requested by the user, the optimizer processes that particular selection criteria first. If an index is available then the transaction is rapid, and if the amount of data retrieved is small the entire retrieval process is fairly quick. Therefore, the first thing that must be done, is ensure that the proper indexes are available to the optimizer. The following elements are used as selection criteria: (1) NODE.NODE_NAME; (2) NODE.NET_ABBR_NAM; (3) NODE.ACT_STATUS'; (4) NETWORK.AGENCY; (5) RESLOC.STATE; and (6) RESLOC.REGION. Of these elements, only NODE_NAME is a key. As a result, retrievals using a specific node name take less than three seconds. On the other hand, retrievals using specific information on the other elements take between 30 and 60 seconds, except for NETWORK.AGENCY which takes less than 15 seconds (this is because NETWORK is a fairly small table, while RESLOC and NODE are large tables). By creating indexes on each of these elements, retrievals fall below four seconds on the average.

The second type of retrieval, retrievals of all data, is a much more complex and time-consuming transaction.

Since there is no restrictive selection criterion for the optimizer to use, it must perform joins on all the tables with large amounts of data. Although the optimizer does not actually perform cartesian products of the tables, it does have a specific algorithm that it performs to process the joins. Since it is an INGRES algorithm, we cannot determine exactly what the optimizer does, but we can observe some of its peculiar effects. One peculiar effect is that the more indexes there are, the longer the process takes. Although creating all those indexes helps improve retrievals of selected data, it can increase the processing time for retrievals of all data from 56.2 seconds to one minute and 33.97 seconds (we are not sure exactly why this happens). Because of this algorithm, we are unable to determine what path the optimizer takes in processing the retrieval. Therefore, a hit-and-miss method is used to try and find a solution to the bottleneck. The following are some methods that result in little or no success: (1) shuffling the selection criteria; (2) changing physical storage structures; (3) creating new indexes; and (4) resorting the base table so that the data is physically adjacent and not just logically adjacent resulting in improved sequential search times. The only method that shows any improvement (only a two second decrease in processing time) is eliminating all indexes. However, this is not a good solution since it increases the processing time dramatically for those

retrievals that use the indexes. Changing the logical structure will not work in this case either. Since all the views use RESLOC, NETWORK, and STATEREG, combining these tables with other tables would create too big a data table and would affect other transactions that use these tables. As a result, we are unable to resolve this particular type of bottleneck.

There are other bottlenecks in this module that are caused by using unnecessary "sort" commands. Two of the procedures that retrieve data from views also use a sort command. The sort that makes the most significant impact is used in the 'mlstpers1' procedure which sorts the output in alphabetical order:

```
retrieve (listpers.pers_name,
          listpers.position,
          etc.)
  where listpers.persname = sel_pers_name
    and (listpers.net_abbr_nam = sel_net_name  or
         listpers.personl_info = temp_net)
    and ... etc.
  sort by pers_name, net_abbr_name
```

Assuming no changes are made to the current retrievals, a transaction time of 39.73 is reduced to 16.63 by eliminating the sort command, a performance improvement of 58%. Because of the access path that the optimizer takes in processing the retrieval, the result is already in alphabetical order. In addition, since there is only one appearance of each person in the table, the second sort on the element NET_ABBR_NAME is also unnecessary, eliminating the need for the sort command. Outputs from retrievals with and without

the sort command are exactly the same. Of lesser significance is the sort used by the procedure mlstnode1 which sorts the output in NET_ABBR_NAM order. In this case the transaction time is reduced from 57.19 seconds to 49.59 seconds, an improvement of only 13%. The outputs are also identical with or without the sort command because the access path that the optimizer takes puts the result in NET_ABBR_NAM order. In these two cases, although the bottleneck is not actually eliminated, the transaction time is reduced without affecting the results of the output.

This module also contains two instances of erroneous coding that, although it technically does not create a bottleneck, prevents the module from processing. Both the procedures mlstlink1 and mentpers1 attempt to retrieve data from a table, LOCATION, that no longer exists in the data base. Research shows that the LOCATION table has been replaced by the tables RESLOC and STATICLOC, and has been erased from the data base. Attempts to process the module that calls either of these two procedures results in an error message saying that the table cannot be found. No attempt to fix this error is made here.

4. Damage Assessment

The only bottleneck in this module appears in two procedures that perform basically the same transaction. The transaction consists of appending data to a number of tables, with each append requiring a join of at least two

tables. Both the 'ddmgres2' and the 'dlstres2' procedures attempt to retrieve data from four tables: (1) OPTRNCNTR; (2) NODE; (3) ASSETCNTR; and (4) PERSONNEL, e.g.:

```
    append to dresrecd (type = "p",
                        resource = personnel.pers_name,
                        network =  personnel.net_abbr_nam,
                        state = resloc.state,
                        status = "CASUALTY",
                        user name = uname)
      where personnel.dmgcnt > 0
        and personnel.latitude = resloc.latitude
        and personnel.longitude = resloc.longitude
```

Since OPTRNCNTR and ASSETCNTR are small tables, appends to them are fairly rapid, averaging under three seconds. However, appends to NODE and PERSONNEL take an average of 21.76 and 17.49 seconds respectively.

This bottleneck is different from previous ones in that one of the selection criteria is a range criterion vice an equality criterion, specifically, all values of the DMGCNT attribute that are greater than zero. Since this attribute appears as a key in indexes for both NODE and PERSONNEL, the optimizer should be able to use the indexes to locate the desired data. However, the INGRES optimizer does not treat the range comparison the same as an equality. Because the optimizer sees the 'greater than' sign instead of an 'equal' sign, the optimizer does not use the indexes but instead performs a sequential search of the base table causing the bottleneck in the processing time. The way to resolve this bottleneck is to force the optimizer to use the

index to make its search. This requires a change in the program code.

The program code currently calls for retrieving data form the NODE and PERSONNEL table. The indexes that are sorted on the element DMGCNT are NODEDMGSTAT and PERSDMG respectively. The program code needs to be changed so that data are retrieved from NODEDMGSTAT and PERSDMG vice NODE and PERSONNEL respectively. In this way, the optimizer is forced to use the indexes to retrieve the data, e.g.:

```
    append to dresrecd (type = "p",
                        resource = persdmg.pers_name,
                        network = persdmg.net_abbr_nam,
                        state = resloc.state,
                        status = "CASUALTY",
                        user name = uname)
      where persdmg.dmgcnt > 0
        and persdmg.latitude = resloc.latitude
        and persdmg.longitude = resloc.longitude
```

By using the index, the optimizer can immediately locate the first value greater than zero and begin retrieving all data starting from that point, resulting in a much quicker access time than a sequential search. One problem with this is that all the data that is required to be retrieved does not reside in the indexes. Since the retrieval is being made from the indexes vice the base table, the index must contain the data to be retrieved. Therefore, the indexes must be expanded to include not only DMGCNT but the additional data required to be retrieved:

index of personnel is persdmg (dmgcnt,latitude,longitude,
pers_name,net_abbr_nam,state)

By making these changes to the index, and changing the code to reflect retrieval from the index vice the base table, the average processing time decreases from 21.76 seconds to 4.21 seconds for the NODE table and from 17.89 seconds to 2.28 seconds for the PERSONNEL table. This represents an 80% and 87% increase in processing efficiency respectively. To improve the processing efficiency for the entire procedure, the same type of changes can be made to the retrievals from OPTRNCNTR and ASSETCNTR reducing their processing time to under two seconds.

5. Service Requests

The only significant bottleneck in this module involves the use of the sort command. However, in this case, the sort command is necessary and, in fact, is the main function of the procedure. The procedure 'ssort1' is called to retrieve data and to sort it into the order determined by the user. Time trials on the retrievals show that without the sort command, transaction time is only 3.04 seconds, an acceptable speed. However, with the sort command, the transaction time jumps to 9.66 seconds. The only way to eliminate the bottleneck is to eliminate the sort command. The only way to eliminate the sort command is to force the optimizer to retrieve data so that the output is already in the order desired.

Forcing the optimizer to take a particular access path will normally require a different index for each different sort format, which can mean quite a lot of indexes. However, an analysis of the structure of the SERVREQUEST table, from which the data is being retrieved, shows that even though there are seven different sort options available to the user, only two indexes will have to be created while five existing indexes will need to be modified. SERVREQUEST uses a binary tree structure and has five indexes already created. The modifications to the existing indexes merely consist of adding additional keys to the indexes. Test runs after the changes are made show that the outputs are sorted in the order desired without the use of the sort statement. By making these changes to the physical structure of the SERVREQUEST table, normal transaction time can be reduced from 9.66 seconds to 3.04 seconds, a performance improvement of 69%. The modification to five indexes and the creation of two indexes, however, raises the question of negative effects resulting from the creation of additional indexes.

E. TRADEOFFS OF PROPOSED SOLUTIONS

The most obvious tradeoff of creating additional indexes is the increase in processing time when adding information to the data tables. Whenever new information is added to the base table, all the associated indexes must also be updated. The more indexes there are, of course, the longer

the process takes. To determine how much of an impact this will cause, time trials are needed to measure the transaction time of adding data before and after the indexes are created. The PERSONNEL table will be used as a test table since it is a large table and will provide a better spread of time measurements.

Prior to adding any indexes to the PERSONNEL table, the processing time for appending data to the table is measured. When data are added to a table, the optimizer looks at the element or elements that are keys to the table. It takes this value and quickly determines where in the table the data are to be stored. Because it is using the keys, and not performing a sequential search, adding data is normally fairly rapid. The optimizer also uses the keys when updating the table indexes. As a result, additions to the data tables are normally faster than retrievals since there is very little sequential searching. In addition, unless the table is incredibly huge and there are a lot of overflow pages, the size of the table does not significantly affect the update time. Since keys are used, the location for storing the new data is quickly determined. The critical factor in update time is the number of indexes that a table has. For every data item that is added to the base table, each index must also be updated. The average processing time for adding data to the PERSONNEL table, without indexes, is 2.5 seconds. After adding one index, the

average processing time changes insignificantly to 2.58 seconds. However, when five indexes are added, the processing time increases to an average of 3.2 seconds. Even though this is a 28% increase in transaction time, the actual time is still small. Since adding indexes can make a significant improvement to the processing efficiency for retrievals, the tradeoff seems to favor adding the index. However, it should be noted that there are other factors which will affect the decision. If the retrieval is already fast, the small decrease in processing time will be offset by the small increase in update time. Therefore, indexes should only be used when a significant improvement can be realized. Also, consideration should be given to how often updates are done compared to retrievals. This will also help determine whether the index should be created or not.

Another tradeoff of creating additional indexes is the additional memory required to store the indexes. The size of the base table determines the amount of memory needed to create the index. However, in most cases, this is an insignificant amount. Although it is up to the user to determine what kind of tradeoffs to make with regard to creating indexes, it is shown here that the negative aspects of additional indexes is minimal and far outweighed by the improvement in processing efficiency.

F. GENERAL IMPROVEMENTS

Improvements in processing efficiency need not be limited to specific changes to the data base structure. INGRES optimization features such as the "optimizedb" command can also be routinely utilized to ensure the optimizer has all the access paths available to it. In addition, compression of data and verification of fill factors should also be periodically performed. These tasks are normally the responsibility of the Data Base Administrator (DBA). Another important tool is the data dictionary. Although a data dictionary in itself does not improve processing efficiency, it provides an excellent means of managing the data tables within the data base. Information on what indexes are available, what the keys are, and what storage structure a table is using are all vital information in determining how to improve the processing efficiency of the program. If this information is not available to the programmer, he will not know when retrievals are inefficient nor how to make them efficient. Consequently, the data dictionary becomes an important tool in improving the processing efficiency of the program.

VII. CONCLUSION

This thesis has attempted to accomplish two major goals: (1) generate documentation that describes the EPMIS data base in detail; and (2) use the documentation to make significant improvements in the processing efficiency of the EPMIS program. The documentation includes: (1) a listing of all data tables within the EPMIS data base grouped by permanent tables, temporary tables, indexes, and views; (2) a table showing the separation of the permanent tables into major groups; (3) an Entity-Relationship Diagram graphically illustrating how the various entities within each group are related; and (4) a listing of the physical storage structure used by each table. Using this documentation, an analysis of the EPMIS program has resulted in proposed changes to the EPMIS data base that reflect significant improvements in processing efficiency. These proposed changes are summarized in Exhibit 10. General recommendations are also made on improving the processing efficiency even further. Implementation of these changes and recommendations can result in an EPMIS program that responds much more rapidly, greatly enhancing its viability in crisis situations. In concluding this effort, some proposals for follow-on projects concerning the EPMIS program are offered.

One possible follow-on project involves a study of the predicted usage of the EPMIS data base, determining how each data table will grow, and using this information to predict potential future bottlenecks in the program. This thesis concentrated on bottlenecks that exist based on the amount of data currently loaded into the data base. During the analysis, however, situations were discovered that could lead to potential bottlenecks if the data tables were to grow significantly. Another project would be to develop a more comprehensive active data dictionary within INGRES. The current INGRES data dictionary provides only a list of all data tables, their elements, and technical information such as physical storage structure and secondary indexes. A more complete data dictionary would not only provide the technical information on each data table, but would also provide a descriptive explanation of what each table is used for, a definition of all the elements in the table, and a cross reference of each table to the procedures that it is used in. In addition, by being an active data dictionary, it would accurately reflect all changes to the data base. The importance of this is shown in Exhibit 11 which lists all the tables that have already been added to the EPMIS data base since this thesis began and therefore are not included in the earlier exhibits or in the E-R diagram. A final project would be to look at the bottlenecks this thesis did not examine, i.e., bottlenecks caused by the

operating system and bottlenecks involving security access validations. These bottlenecks still exist in the current EPMIS program and may be a source of significant delays in program processing.

APPENDIX

EXHIBITS

The following exhibits describe the logical and physical structure of the EPMIS data base, and provide a list of processing bottlenecks as well as methods for resolving them.

EPMIS DATABASE

40 Permanent Tables:

agcyfunct	asset	assetcntr	baddam *
carriers	claimno	direction	dlaylist *
dmgedres	dmgobsrvd	eadlist	eads
facltycom	facltyreq	facstat	functmap
jdmgobsrvd	journcomment	journrequest	jrnfaccom
jrnfacreq	laydown	link	location *
network	node	optrncntr	personnel
perstatus	rectangle	reqstatus	resloc
servcomment	servrequest	sonsit	statereg
staticloc	statnatn	telords *	tsprpmap

* no longer being used

25 Temporary Tables:

acrecd	asstrecd	dcirrecd	dlayrecd
dlistrecd	drectrecd	dresrecd	faccomrpt
facreqrpt	freqrecd	linkrecd	netrecd
nmaprecd	noderecd	ocrecd	persrecd
pocrecd	sreqrecd	srvcomrpt	srvreqrpt
teadlrecd	teadrecd	temprec2	temprecall
tempreprecal			

19 Indexes:

Index	Indexed on	Index	Indexed on
dmgeloc	dmgedres	dmgodir	dmgobsrvd
dmgosys	dmgobsrvd	dmgotyp	dmgobsrvd
loclocsta	location	locstatus	location
locstcty	location	nodeloc	node
nodenodnet	node	persloc	personnel
persnet	personnel	persstat	personnel
servagcy	servrequest	servcir	servrequest
servnccag	servrequest	servpri	servrequest
servstcar	servrequest	stlatlon	statereg
streg	statereg		

6 Views:

listacs	listassets	listnet
listnodes	listocs	listpers

Exhibit 1

DATA TABLES BROKEN DOWN BY GROUPS

COMMUNICATION NETWORKS

asset	assetcntr	link	network *
node	optrncntr	personnel	perstatus
resloc *	staticloc		

DAMAGE ASSESSMENT

direction	dmgedres	dmgobsrvd	jdmgobsrvd
laydown	rectangle	resloc	statereg

SERVICE REQUESTS

agcyfunct *	carriers	functmap	journcomment
journrequest	reqstatus	servcomment	servrequest
tsprpmap *			

FACILITY REQUESTS

agcyfunct	claimno	facltycom	facltyreq
facstat	jrnfaccom	jrnfacreq	network
tsprpmap			

EMERGENCY ACTION DOCUMENTS

eadlist	eads

REGIONAL SITUATION

statnatn	sonsit

* Tables that appear in more than one group

Exhibit 2

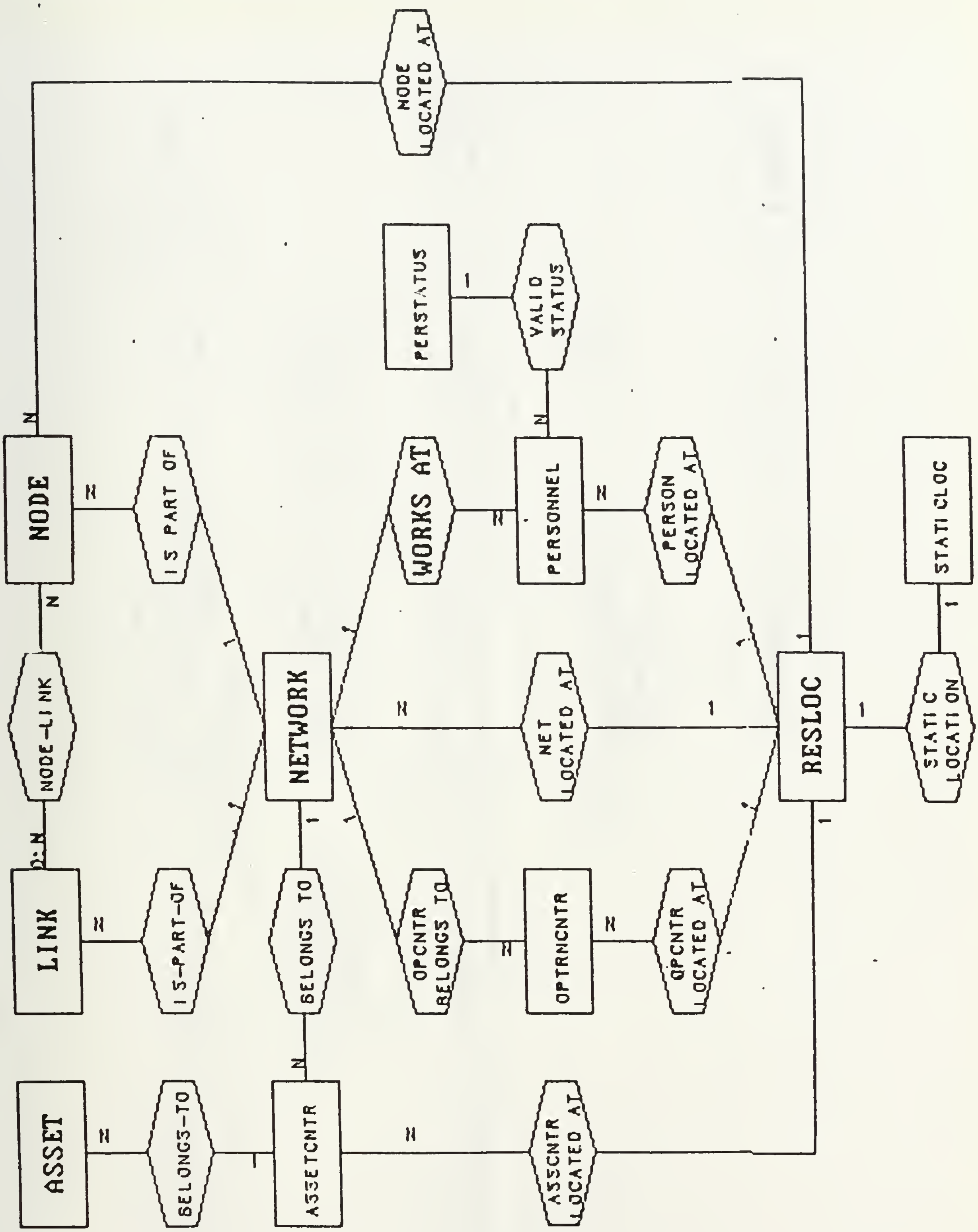

NODE
LINK
ASSET
NETWORK
RESLOC
PERSONNEL
OPTRNCNTR
ASSETCNTR
PERSTATUS
STATICLOC
NODE LOCATED AT
VALID STATUS
IS PART OF
WORKS AT
PERSON LOCATED AT
NODE-LINK
NET LOCATED AT
STATIC LOCATION
IS-PART-OF
BELONGS TO
OPCNTR BELONGS TO
OPCNTR LOCATED AT
BELONGS-TO
ASSCNTR LOCATED AT

Exhibit 3

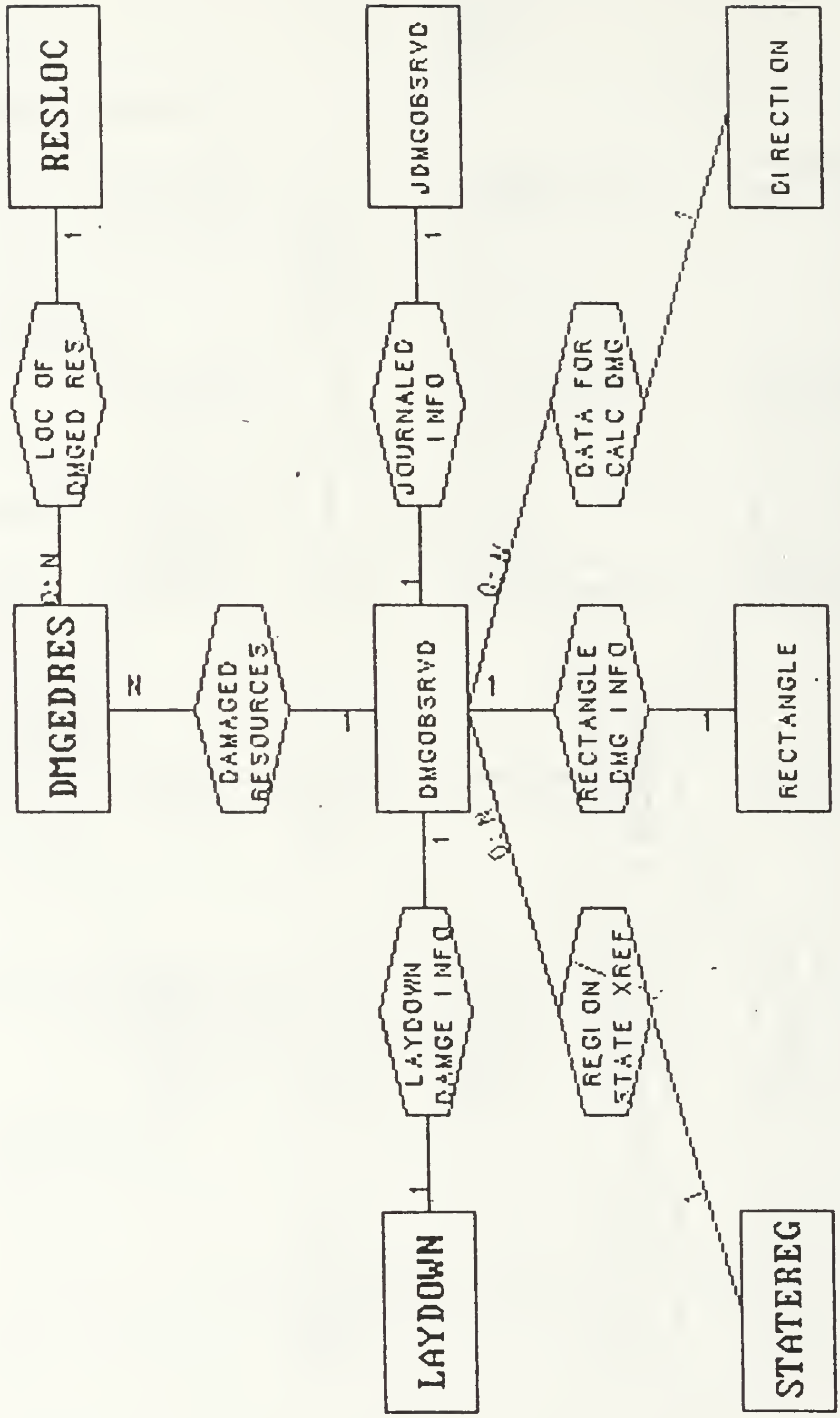
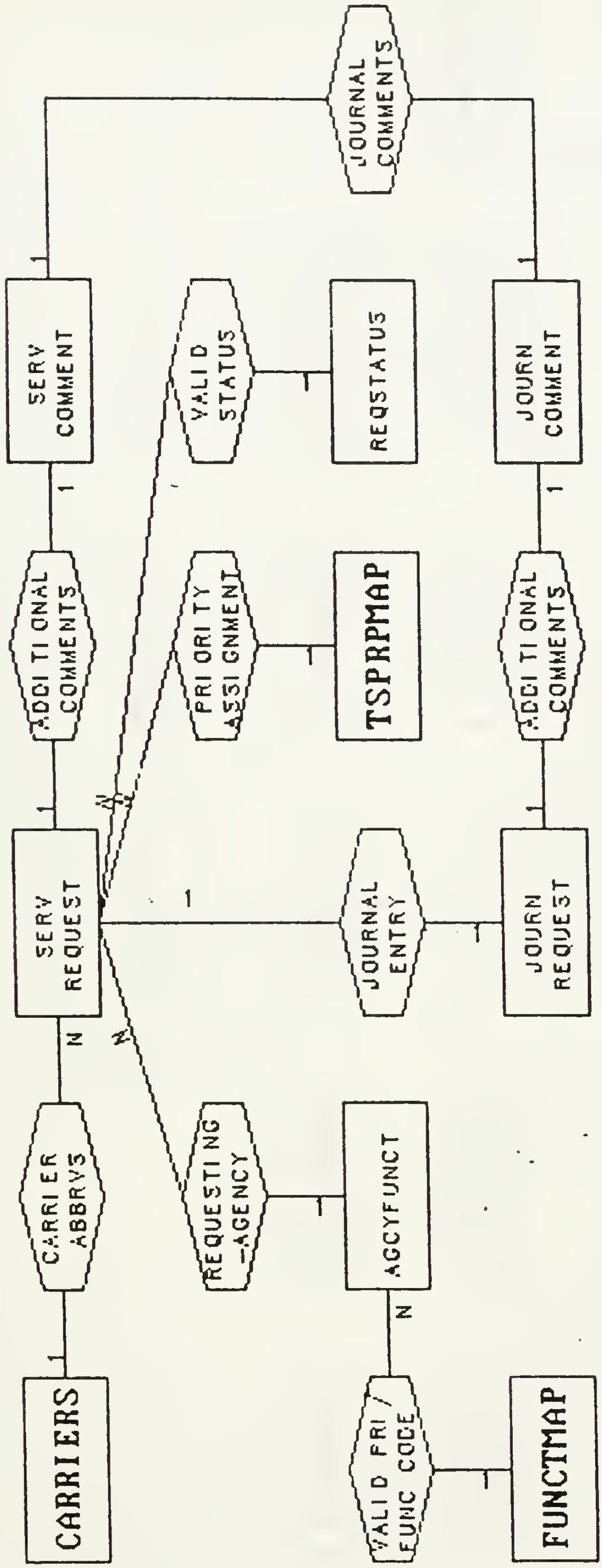

RESLOC
LOC OF DMGED RES
DMGEDRES
DAMAGED RESOURCES
JDMGOBSRVD
JOURNALED INFO
DMGOBSRVD
DIRECTION
DATA FOR CALC DMG
RECTANGLE
RECTANGLE DMG INFO
LAYDOWN RANGE INFO
LAYDOWN
REGION/STATE XREF
STATEREG

Exhibit 4



CARRIERS
CARRIER ABBRVS
SERV REQUEST
ADDITIONAL COMMENTS
SERV COMMENT
JOURNAL COMMENTS
JOURN COMMENT
VALID STATUS
REQSTATUS
PRIORITY ASSIGNMENT
TSPRPMAP
JOURNAL ENTRY
JOURN REQUEST
REQUESTING_AGENCY
AGCYFUNCT
VALID FRI / FUNC CODE
FUNCTMAP

Exhibit 5

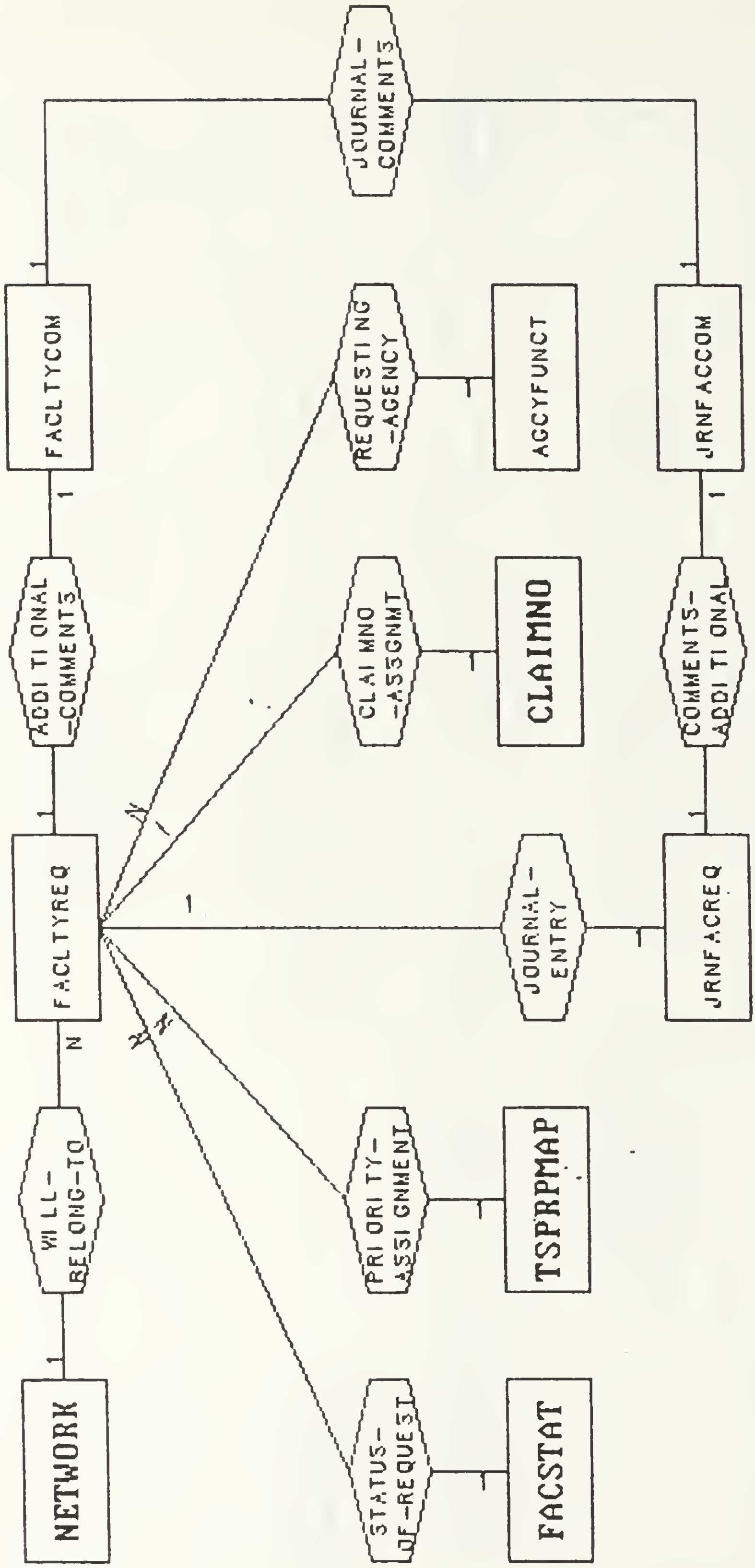

NETWORK
WILL-BELONG-TO
FACLTYREQ
ADDITIONAL-COMMENTS
FACLTYCOM
JOURNAL-COMMENTS
REQUESTING-AGENCY
AGCYFUNCT
CLAIMNO-ASSGNMT
CLAIMNO
JOURNAL-ENTRY
JRNFACREQ
COMMENTS-ADDITIONAL
JRNFACCOM
PRIORITY-ASSIGNMENT
TSPRPMAP
STATUS-OF-REQUEST
FACSTAT

Exhibit 6

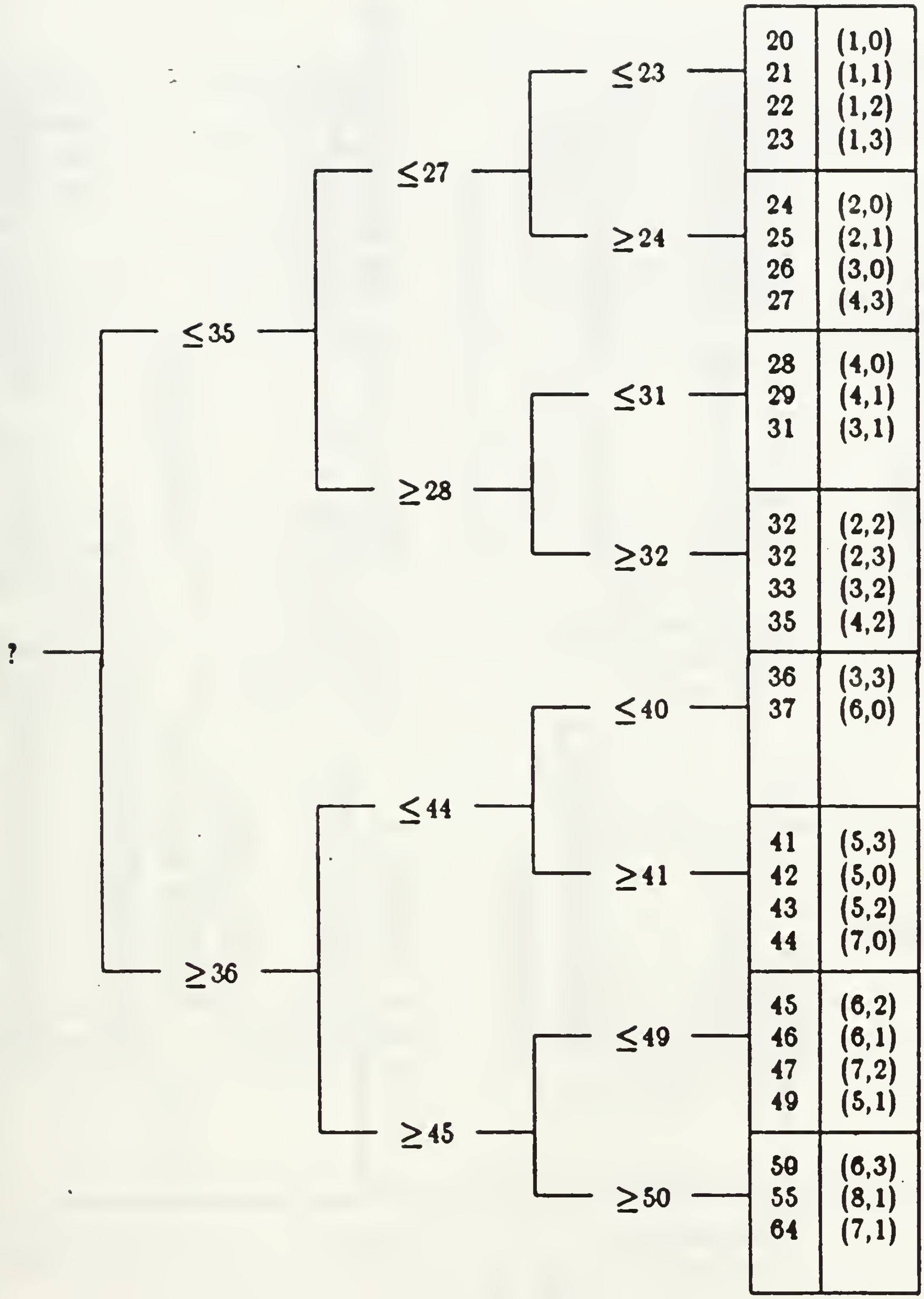

?
≤35
≥36
≤27
≥28
≤44
≥45
≤23
≥24
≤31
≥32
≤40
≥41
≤49
≥50
20 (1,0)
21 (1,1)
22 (1,2)
23 (1,3)
24 (2,0)
25 (2,1)
26 (3,0)
27 (4,3)
28 (4,0)
29 (4,1)
31 (3,1)
32 (2,2)
32 (2,3)
33 (3,2)
35 (4,2)
36 (3,3)
37 (6,0)
41 (5,3)
42 (5,0)
43 (5,2)
44 (7,0)
45 (6,2)
46 (6,1)
47 (7,2)
49 (5,1)
50 (6,3)
55 (8,1)
64 (7,1)

Exhibit 7

List of Tables by Storage Structure

HEAP

Acrecd(TT)	Assetcntr	Asstrecd(TT)	Baddam
Claimno	Dcirrecd(TT)	Dlaylist	Dlayrecd(TT)
Dlistrecd(TT)	Drectrecd(TT)	Dresrecd(TT)	Eadlist
Eads	Facreqrpt(TT)	Facstat	Functmap
Laydown	Optrncntr	Perstatus	Reqstatus
Sonsit	Temprecall(TT)	Tempreprecal(TT)	
Srvreqrpt(TT)	Freqrecd(TT)	Jdmgobsrvd	Linkrecd(TT)
Netrecd(TT)	Nmaprecd(TT)	Noderecd(TT)	Ocrecd(TT)
Persrecd(TT)	Pocrecd(TT)	Sreqrecd(TT)	Statnatn
Teadlrecd(TT)	Teadrecd(TT)	Telords	Temprec2(TT)
Tsprpmap			

HASH

Agcyfunct	Asset	Carriers	Direction
Dmgedres	Dmgeloc(SI)	Dmgosys(SI)	Dmgotyp(SI)
Faccomrpt(TT)	Facltycom	Facltyreq	Journcomment
Journreqest	Jrnfaccom	Jrnfacreq	Link
Location	Locstatus(SI)	Network	Nodeloc(SI)
Persloc(SI)	Rectangle	Servcomment	Srvcomrpt(TT)
Statereg	Streg(SI)	Staticloc	Resloc

ISAM

Servpri(SI)	Stlatlon(SI)

BINARY TREE

Dmgobsrvd	Dmgodir(SI)	Loclocsta(SI)	Locstcty(SI)
Node	Nodenodnet(SI)	Persnet(SI)	Personnel
Persstat(SI)	Servagcy(SI)	Servcir(SI)	Servnccag(SI)
Servrequest	Servstcar(SI)		

VIEWS

Listacs	Listassets	Listnet	Listnodes
Listocs	Listpers		

TT - Temporary Table
SI - Secondary Index

Exhibit 8

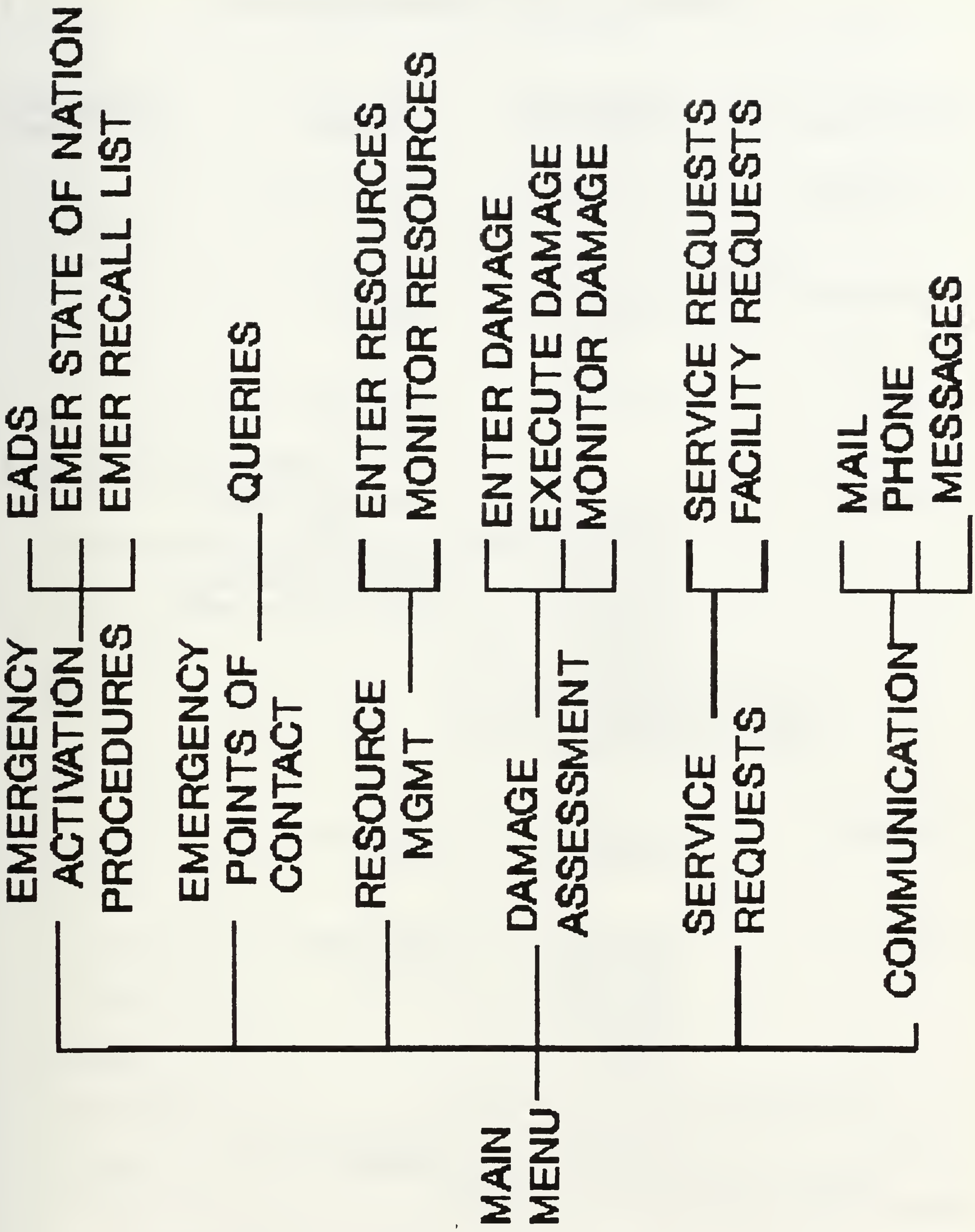

MAIN MENU
EMERGENCY ACTIVATION PROCEDURES
EADS
EMER STATE OF NATION
EMER RECALL LIST
EMERGENCY POINTS OF CONTACT
QUERIES
RESOURCE MGMT
ENTER RESOURCES
MONITOR RESOURCES
DAMAGE ASSESSMENT
ENTER DAMAGE
EXECUTE DAMAGE
MONITOR DAMAGE
SERVICE REQUESTS
SERVICE REQUESTS
FACILITY REQUESTS
COMMUNICATION
MAIL
PHONE
MESSAGES

Exhibit 9

SUMMARY OF BOTTLENECKS AND SOLUTIONS

Tranasaction	Procedures Affected	Solution	Efficiency Improvement
1. retrieve	erecall1 egetrcrp1	create new index	91 %
2. retrieve	pemgypoc1	add new element	82 %
3. sort	pemgypoc1	eliminate sort command	37 %
4. delete	uvalidloc mupdpers1 mupdoc1 mupdnode1 mupdnet1 mupdac1 maddpers1 maddoc1 maddnode1 maddac1 maddnet1	create new index and modify physical storage structure from isam to hash	61 %
5. append	rcombine mcombine	create new index	65 %
6. join	mlstpers mlstasst1 mlstnet1 mlstnode1 mlstoc1 mlstac1	create new indexes	87-93 %
7. sort	mlstpers1	eliminate sort command	58 %

Exhibit 10

8. sort	mlstnodel	eliminate sort command	13 %
9. append	ddmgres2 dlstres2	create new index and modify program code	87 %
10. sort	ssortl	create new indexes, modify old indexes, eliminate sort command	69 %

<u>Summary of new indexes:</u>

Base Table	Element(s)
personnel	poc_recldby
personnel	epoc
resloc	lat_degrees, lat_minutes, lat_seconds, lon_degrees, lon_minutes, lon_seconds
resloc	city, state
node	net_abbr_nam
node	act_status
network	agency
resloc	state
resloc	region
personnel	dmgcnt, latitude, longitude, pers_name, net_abbr_nam, state
node	dmgcnt, dmgstatus, latitude, longitude, node_name, net_abbr_nam
servrequest	agency, priority, datetime

Exhibit 10

servrequest	carrier_name, priority, datetime
servrequest	ncc_number, priority, datetime
servrequest	priority, datetime
servrequest	datetime, priority
servrequest	circuit_num, priority, datetime
servrequest	status, priority, datetime

Exhibit 10

RECENTLY ADDED DATA TABLES

Permanent Tables:

classify	combloc	ddelement	ddgroup
ddgrpelt	ddlink	ddlinks	ddtemp
ddtypeconv	dmgeloc	eerselect	graphics
low_map	nodex	person	probdmge
repclass	status_trans	tempgrp	vntk
weapon	wild	xassetcntr	xoptrncntr

Temporary Tables:

dmgrprecd	dtmprcd	mapnodes	tempelt
templink	twponrecd	stat_convert	

Indexes:

dmgelatlon	dmgexflg	dmgoexec	dmgolatlon
drestyp	jdmglatlon	nodedmgstat	nodelatlon
persdmg	perslatlon	probminmax	xlpersonnel

Views:

damage	dmgodir

Exhibit 11

LIST OF REFERENCES

1. Booz-Allen & Hamilton Inc., "Emergency Preparedness Management Information System (EPMIS) Five Year Plan (Draft)," 19 September 1988.

2. National Communications System, Office of Emergency Preparedness, "Emergency Preparedness Management Information System (EPMIS) Program Plan," 07 April 1986.

3. Booz-Allen & Hamilton Inc., "Emergency Preparedness Management Information System (EPMIS) Deployment Plan," 19 September 1988.

4. Chen, P., The Entity-Relationship Approach to Logical Data Base Design, QED Information Sciences, Inc., 1977.

5. Relational Technology Inc., INGRES Training Manual, QUEL Course Version, Overview--Introduction to INGRES, 1 August 1987.

6. Relational Technology Inc., INGRES Training Manual, QUEL Course Version, Performance Part I Storage Structures, 1 August 1987.

7. Kroenke, D.M. and Dolan, K.A., Database Processing: Fundamentals, Design, Implementation, 3d ed., Science Research Associates Inc., 1988.

8. Date, C.J., A Guide to INGRES, Addison-Wesley Publishing Co. Inc., 1987.

9. Relational Technology Inc., INGRES Training Manual, QUEL Course Version, Performance Part II Storage Structures, 1 August 1987.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	LCDR William B. Short 1705 NW Viewmont Court Silverdale, Washington 98383	1
4.	LT Jeffrey M. Bockenek Pacific Operations Support Facility Box 9 Pearl Harbor, Hawaii 96860-7150	1
5.	Professor Daniel R. Dolk, Code 54Dk Naval Postgraduate School Monterey, California 93943-5000	1
6.	Prof. Barry Frew, Code 54Fw Naval Postgraduate School Monterey, California 93943-5000	1
7.	Professor Magdi Kamel, Code 54Ka Naval Postgraduate School Monterey, California 93943-5000	2
8.	Mr. Norman Douglas Program Manager--EPMIS National Communications System 8th & South Courthouse Road Arlington, Virginia 22204	2
9.	Mr. Jay Roland Rolands & Associates Corporation 500 Sloat Avenue Monterey, California 93940	1